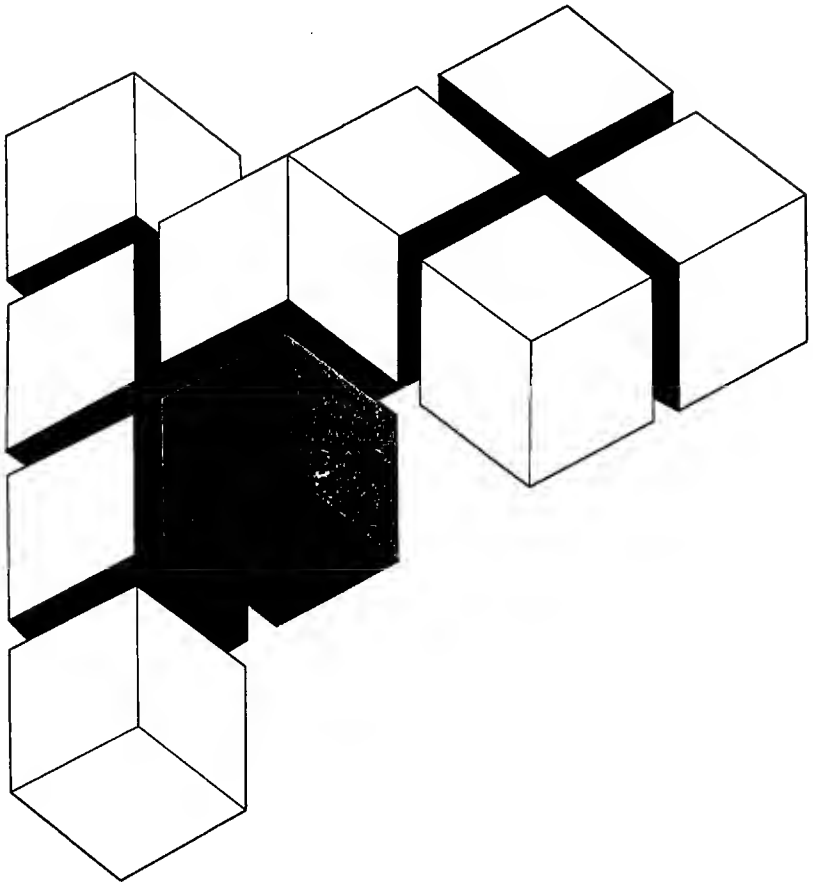
LY28-1308-4

TSO Extensions
Terminal Monitor Program and Service Routines



TSO Extensions
Terminal Monitor Program and Service Routines

LY28-1308-4

Fifth Edition (September, 1987)

This is a major revision of, and obsoletes, LY28-1308-3. See the Summary of Changes following the Contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Release 3 of TSO Extensions Program Number 5665-285 and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product in this publication is not intended to state or imply that only IBM's product may be used. Any functionally equivalent product may be used instead.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department D58, Building 921-2, PO Box 390, Poughkeepsie, N.Y. 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This publication describes the Time Sharing Option/Extensions (TSO/E) terminal monitor program (TMP) and the TSO/E service routines.

Who Should Read This Book?

The publication is for people who diagnose TSO/E problems or maintain TSO/E. Usually, this is a system programmer.

The level of detail at which this book is written assumes that the reader:

Can code Job Control Language (JCL) statements to execute programs or cataloged procedures.

Can code in assembler language and can read assembler, loader, and linkage editor output.

Can use system messages, system dumps, and IBM publications, such as *Diagnostic Techniques*, to locate errors in problem programs.

How is This Book Organized?

This book contains an index and the following chapters:

Chapter 1. Introduction describes the purpose and use of the TMP and the TSO service routines and their relationship to the system. You should understand the "Introduction" before reading any of the other chapters.

Chapter 2. Terminal Monitor Program describes the terminal monitor program and its relationship to other programs, including the LOGON/LOGOFF scheduler, the TSO command processors, and the TSO service routines.

Chapter 3. Terminal I/O Service Routines describes STACK, PUTLINE, GETLINE, and PUTGET and their relationship to other programs, including the TMP and the TSO command processors.

Chapter 4. Command Scan and Parse Service Routines describes command scan and parse and their relationship to other programs, including the TMP and the TSO command processors.

Chapter 5. Dynamic Allocation Interface Routine describes the dynamic allocation interface routine (DAIR) and its relationship to other programs.

Chapter 6. Default Service Routine describes the default routine and its relationship to other programs, including PUTLINE, PUTGET, and the Catalog Information Routine.

Chapter 7. Catalog Information Routine describes the catalog information routine and its relationship to other programs, including the routines invoked by the LOCATE macro instruction.

Chapter 8. DAIR/SVC99 Error Code Analyzer describes the DAIR error code analyzer (DAIRFAIL) and its relationship to other programs.

Chapter 9. TSO Message Issuer (IKJEFF02) describes the TSO message issuer and its relationship to other programs.

Chapter 10. TSO Service Facility describes the TSO service facility and its relationship to other programs.

Chapter 11. TSO Service Linkage Assist Routine describes the linkage assist routine and its relationship to other programs.

Chapter 12. CLIST Attention Facility describes the CLIST attention facility and its relationship to other programs.

Chapter 13. CLIST Processing and Diagnosis describes CLIST Phase 1 and Phase 2 processing and the diagnostic information provided in a dump. This includes information for the EXEC command processor and the PUTGET service routine.

How Do You Use This Book?

If you have never used this book, look over the Table of Contents and each chapter to become familiar with the book's content and method of presentation.

To diagnose a problem that occurs when you use TSO/E, start with the procedure in *TSO Extensions System Diagnosis: Guide and Index*. If that procedure leads you to suspect an error in the TMP or in a TSO/E service routine, use this book to further isolate the cause of the problem.

TSO Extensions System Diagnosis: Guide and Index tells you how to report any problem to IBM that you are unable to solve.

Where Can You Find Additional Information?

Additional information is available in the following publications:

TSO Extensions User's Guide, SC28-1333.

TSO Extensions Command Reference, SC28-1307.

TSO Extensions Customization, SC28-1136.

TSO Extensions Programming Guide, SC28-1363.

TSO Extensions Programming Services, SC28-1364.

MVS/Extended Architecture Catalog Diagnostic Guide, SY26-3899.

MVS/Extended Architecture Message Library: System Codes, GC28-1157.

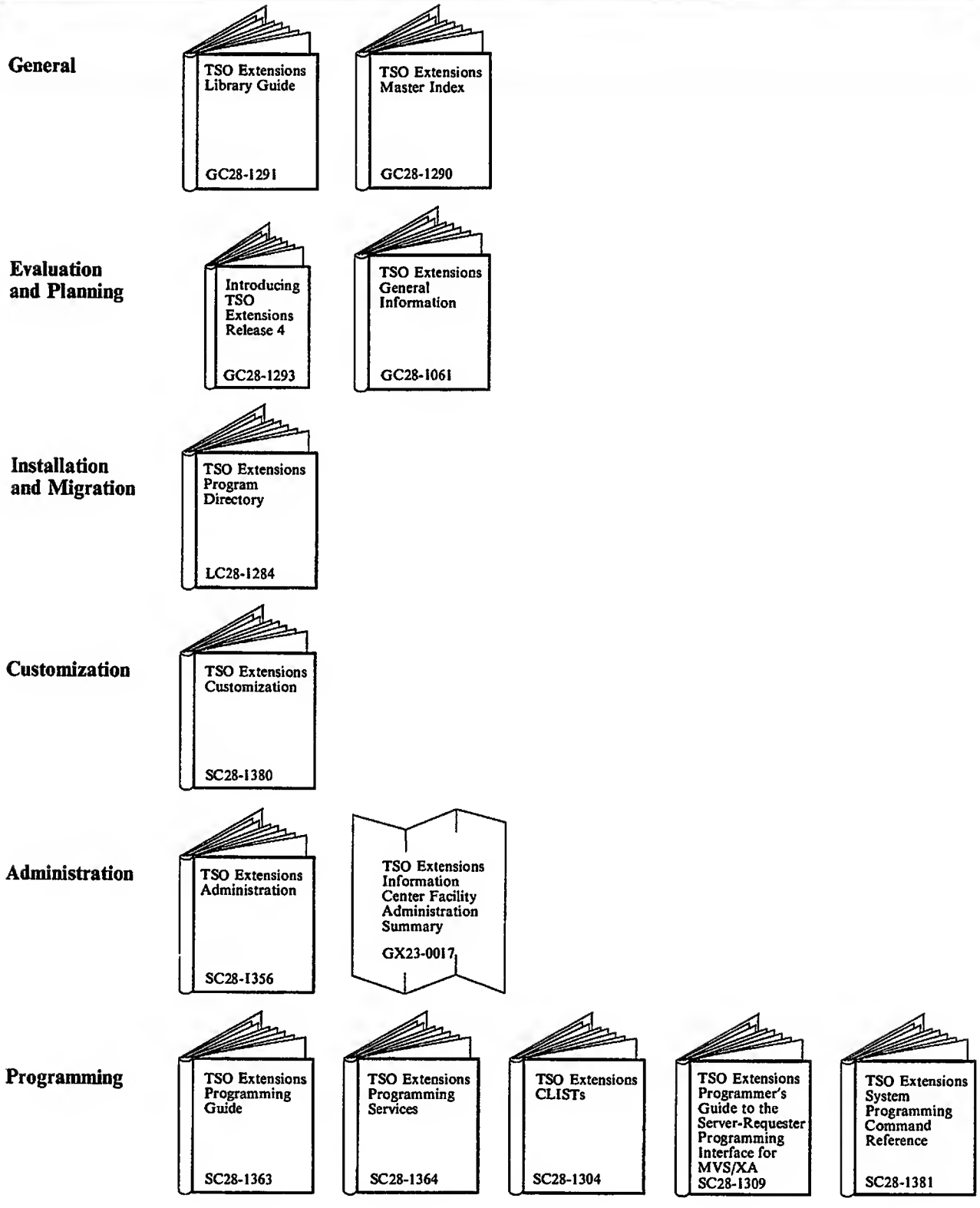
MVS/Extended Architecture System Programming Library: 31-Bit Addressing, GC28-1158

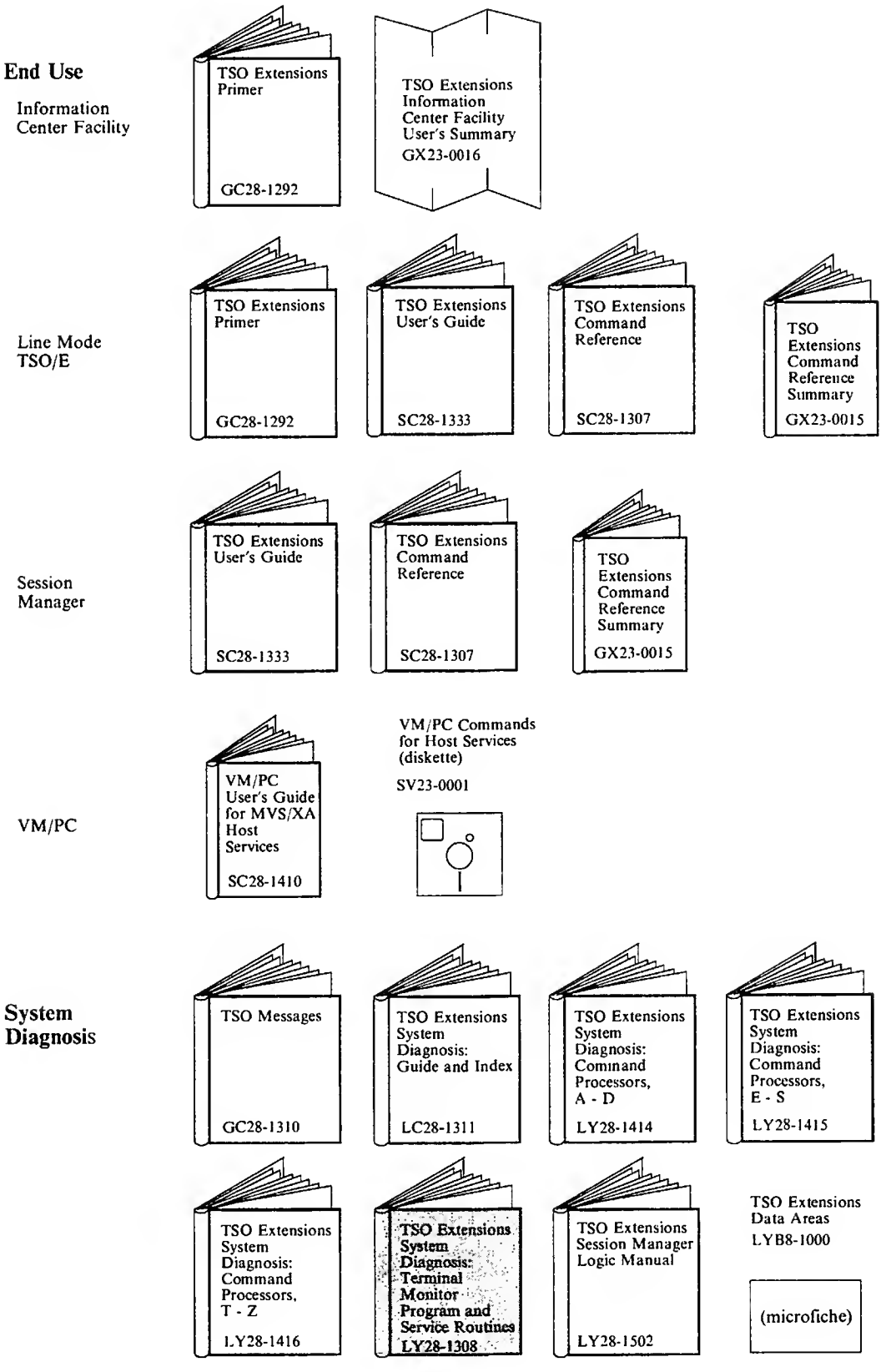
MVS/Extended Architecture Data Areas (microfiche), LYB8-1000.

Do You Have Problems, Comments, or Suggestions?

Your suggestions and ideas can contribute to the quality and the usability of this book. If you have problems using this book, or if you have suggestions for improving it, complete and mail the Reader's Comment Form at the back of the book.

The TSO Extensions Library





Contents

Chapter 1. Introduction 1-1
TSO/E System 1-1
MVS/Extended Architecture Considerations 1-2
TSO Extensions Considerations 1-3
Terminal Monitor Program 1-4
TSO/E Command Processors and User Programs 1-4
TSO/E Service Routines 1-5
Terminal I/O Service Routines 1-5
Command SCAN and PARSE Service Routines 1-5
Dynamic Allocation Interface Routine 1-5
Default Service Routine and Catalog Information Routine 1-5
DAIR Error Code Analyzer 1-5
TSO/E Service Routine 1-6

Chapter 2. Terminal Monitor Program 2-1
Method of Operation 2-3
Diagram 1. Primary Terminal Monitor Program 2-4
Diagram 2. TMP Initialization 2-6
Diagram 3. Handling TSO Commands with the Primary TMP 2-8
Diagram 4. Handling TSO Commands with a Parallel TMP 2-10
Diagram 5. Handling Attention Requests 2-12
Diagram 6. Handling ESTAI Requests 2-14
Diagram 7. Handling ESTAE Requests 2-16
Program Organization 2-18
Program Hierarchy 2-18
Control Flow 2-19
Module Operation 2-22
Data Areas 2-30
Directory 2-31
Diagnostic Aids 2-33

Chapter 3. Terminal I/O Service Routines 3-1
Method of Operation 3-3
Diagram 8. Terminal I/O Service Routines (Overview) 3-4
Program Organization 3-6
Program Hierarchy 3-6
Diagnostic Aids 3-8
I/O Services Modules 3-11

Chapter 4. Command Scan and Parse Service Routines 4-1
Method of Operation 4-3
Diagram 9. Command Scan and Parse Service Routines (Overview) 4-4
Diagram 10. Command Scan Service Routine 4-6
Diagram 11. Parse Service Routine 4-8
Diagram 12. Parse Initialization 4-10
Diagram 13. Searching for IKJPARSR Positional Parameters 4-12
Diagram 14. Searching for IKJPARS2 Positional Parameters 4-14
Diagram 15. Searching for Keyword Parameters and Subfields 4-16
Program Organization 4-18
Program Hierarchy 4-18
Module Operation 4-19

Directory 4-21
Diagnostic Aids 4-25

Chapter 5. Dynamic Allocation Interface Routine 5-1
Method of Operation 5-3
Entry to DAIR 5-3
Diagram 16. Dynamic Allocation Interface Routine 5-4
Program Organization 5-6
Program Hierarchy 5-6
Module Operation 5-6
Directory 5-7
Diagnostic Aids 5-8

Chapter 6. Default Service Routine 6-1
Method of Operation 6-2
Searching The System Catalog 6-3
Exit From Default 6-3
Diagram 17. Default Service Routine 6-4
Program Organization 6-6
Directory 6-6
Diagnostic Aids 6-7

Chapter 7. Catalog Information Routine 7-1
Method of Operation 7-3
Diagram 18. Catalog Information Routine 7-4
Program Organization 7-6
Directory 7-6
Diagnostic Aids 7-7
Catalog Information Routine 7-7
Locate 7-7

Chapter 8. DAIR/SVC99 Error Code Analyzer 8-1
Method of Operation 8-3
Diagram 19. Analyzing DAIR/SVC99 Error Codes 8-4
Program Organization 8-6
Program Hierarchy 8-6
Module Operation 8-6
Directory 8-7

Chapter 9. TSO/E Message Issuer (IKJEFF02) 9-1
Method of Operation 9-3
Diagram 20. TSO Message Issuer 9-4
Program Organization 9-6
Program Hierarchy 9-6
Module Operation 9-6
IKJEFF02 -- TSO/E Message Issuer 9-6
Directory 9-7

Chapter 10. TSO Service Facility 10-1
Diagram 21. TSO Service Facility 10-2
Program Organization 10-4
Program Hierarchy 10-4
Module Operation 10-5
Directory 10-7
Diagnostic Aids 10-8

Chapter 11. TSO/E Service Linkage Assist Routine 11-1
Method of Operation 11-3
Diagram 22. TSO Service Linkage Assist Routine 11-4
Program Organization 11-6
Program Hierarchy 11-6
Module Operation 11-6
Directory 11-7
Diagnostic Aids 11-8

Chapter 12. CLIST Attention Facility 12-1
Method of Operation 12-3
Diagram 23. CLIST Attention Facility 12-4
Program Organization 12-14
Program Hierarchy 12-14
Module Operation 12-14
Directory 12-16
Diagnostic Aids 12-17

Chapter 13. CLIST Processing and Diagnosis 13-1
CLIST Processing Overview 13-1
CLIST Phase 1 Processing 13-1
CLIST Phase 2 Processing 13-2
CLIST Diagnosis Information 13-3
SDWA Contents 13-3
SDWAVRA Contents -- Phase 1 13-3
SDWAVRA Contents -- Phase 2 13-3
CLIST Dump Processing 13-4
CLIST Footprint Description 13-5
CLIST Return Codes 13-5
Services Used by CLIST Processing 13-5

Index X-1

Figures

2-1. TMP Control Flow for a Non-APF-authorized Command Processor or Program 2-20
2-2. Parallel TMP Control Flow for an APF-authorized Command Processor or Program 2-21
13-1. Footprint Data Area 13-5

Summary of Changes

**Summary of Changes
for LY28-1308-4
TSO Extensions (TSO/E) Release 4**

This publication documents the following changes to TSO Extensions (TSO/E) Release 4 in an MVS/XA environment:

- The Command Scan and Parse and the Terminal I/O Service Routine chapters have been updated to reflect enhancements made to TSO/E service routines.
- CLIST processing information is reorganized and presented in a new chapter.

**Summary of Changes
for LY28-1308-3
TSO Extensions (TSO/E) Release 3**

This publication documents the following change to TSO Extensions (TSO/E) Release 3 in an MVS/XA environment:

- CLIST attention facility added. (The CLIST attention facility requires that MVS/System Product Version 2 Release 2.0 be installed.)

**Summary of Changes
for LY28-1308-2
TSO Extensions (TSO/E) Release 3**

This publication documents the following changes to TSO Extensions (TSO/E) Release 3 in an MVS/XA environment:

- LISTDSI CLIST statement added.
- Enhancements made that enable the TSO service facility to handle CLISTs.

In addition, minor technical corrections and editorial changes have been made throughout the book.

As supplied by IBM, the majority of routines comprising the TMP reside in SYS1.ELPALIB and contain logic to execute TSO commands in the background. A definition of "TSO in the background" can be derived from the following table, which shows external and internal indications for foreground, background, TSO and non-TSO:

	FOREGROUND	BACKGROUND
EXTERNAL	TMP is initiated at the terminal	TMP is run by an initiator.
INTERNAL	ASCBTSB = pointer to TSB associated with this TSO user. ECTBKGRD = '0'B	ASCBTSB = 0 ECTBKGRD = '1'B

	TSO	NON-TSO
EXTERNAL	TMP is present.	No TMP is present.
INTERNAL	JSCBPSCB = pointer to the PSCB.	JSCBPSCB = 0

Method of Operation

This section uses the following diagrams to describe the TMP's method of operation:

- Diagram 1: Primary Terminal Monitor Program -- shows the basic functions the TMP performs.
- Diagram 2: TMP Initialization -- shows how the TMP completes the logon process by setting up tables and control blocks that define the user's environment.
- Diagram 3: Handling Commands with the Primary Terminal Monitor Program -- shows how the TMP obtains commands from the terminal and gives control to the appropriate command processor.
- Diagram 4: Handling Commands with a Parallel Terminal Monitor Program -- shows how the parallel TMP obtains commands from the Primary TMP and gives control to the appropriate command processor.
- Diagram 5: Handling Attention Requests -- shows how the TMP handles terminal attention requests.
- Diagram 6: Handling ESTAI Requests -- shows how the TMP attempts to recover from errors in a command processor or one of its programs.
- Diagram 7: Handling ESTAE Requests -- shows how the TMP attempts to recover from errors in its own programs.

Each diagram includes a cross-reference to help you find the appropriate assembly listing.

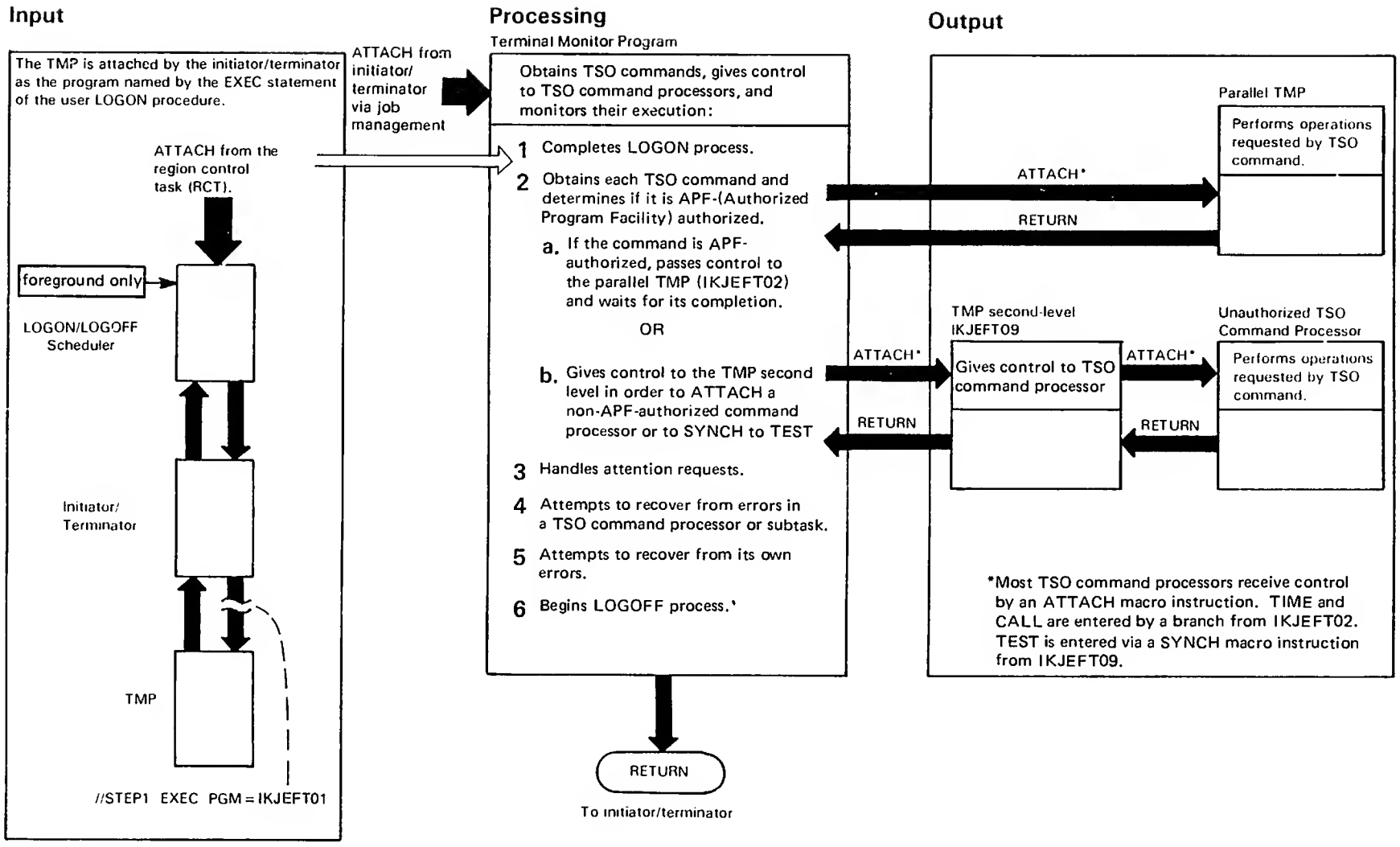


Diagram 1. Primary Terminal Monitor Program (Part 1 of 2)

Key Description	Routine	Label	Diagram
1 Attaches the TMP as the program named by the first EXEC statement in the user logon procedure. The TMP completes the logon process by using PARMLIB tables or setting up tables and control blocks that define the user's environment. Sets up the TMP work areas, displays broadcast messages, and passes control to IKJEFTSC to attach IKJEFT02.	Initialization	IKJEFT01	2
2 Obtains each TSO command or program and determines if it is APF (Authorized Program Facility) authorized. Authorized commands are listed in module IKJEFTE2 and authorized programs in modules IKJEFTE8 and IKJEFTAP.	Mainline	IKJEFT02	3
a If the command is APF-authorized, passes control to a parallel TMP (IKJEFT02) (see Diagram 4 for additional information concerning the parallel TMP) and waits for its completion.			
OR			
b Attaches the TMP second-level routine (IKJEFT09) to give control to the non-APF-authorized command processors. Enters TEST via a SYNCH macro from the TMP second-level routine. Attaches other non-APF-authorized command processors as subtasks of the TMP second-level routine.	Mainline TMP second-level	IKJEFT02 IKJEFT09	4
3 The TMP handles attention interrupts by displaying second-level messages in response to a question mark or by obtaining a new command to replace the one that was interrupted.	Attention Exit Routine	IKJEFT03	5
4 The TMP initiates recovery from errors in a command processor or one of its subtasks by allowing the user to enter a TEST command.	ESTAI Exit Routine	IKJEFT04	6
5 The TMP initiates recovery from errors in its own code by diagnosing the cause of the error and, if possible, by restarting the TMP.	ESTAE Exit Routine ESTAE Retry Routine	IKJEFT05 IKJEFT07	7 7
6 The TMP performs cleanup operations and returns to the initiator/terminator when the operator issues a STOP or MODIFY command or when the user issues a LOGON or LOGOFF command.	Mainline Initialization	IKJEFT02 IKJEFT01	3 2

Diagram 1. Primary Terminal Monitor Program (Part 2 of 2)

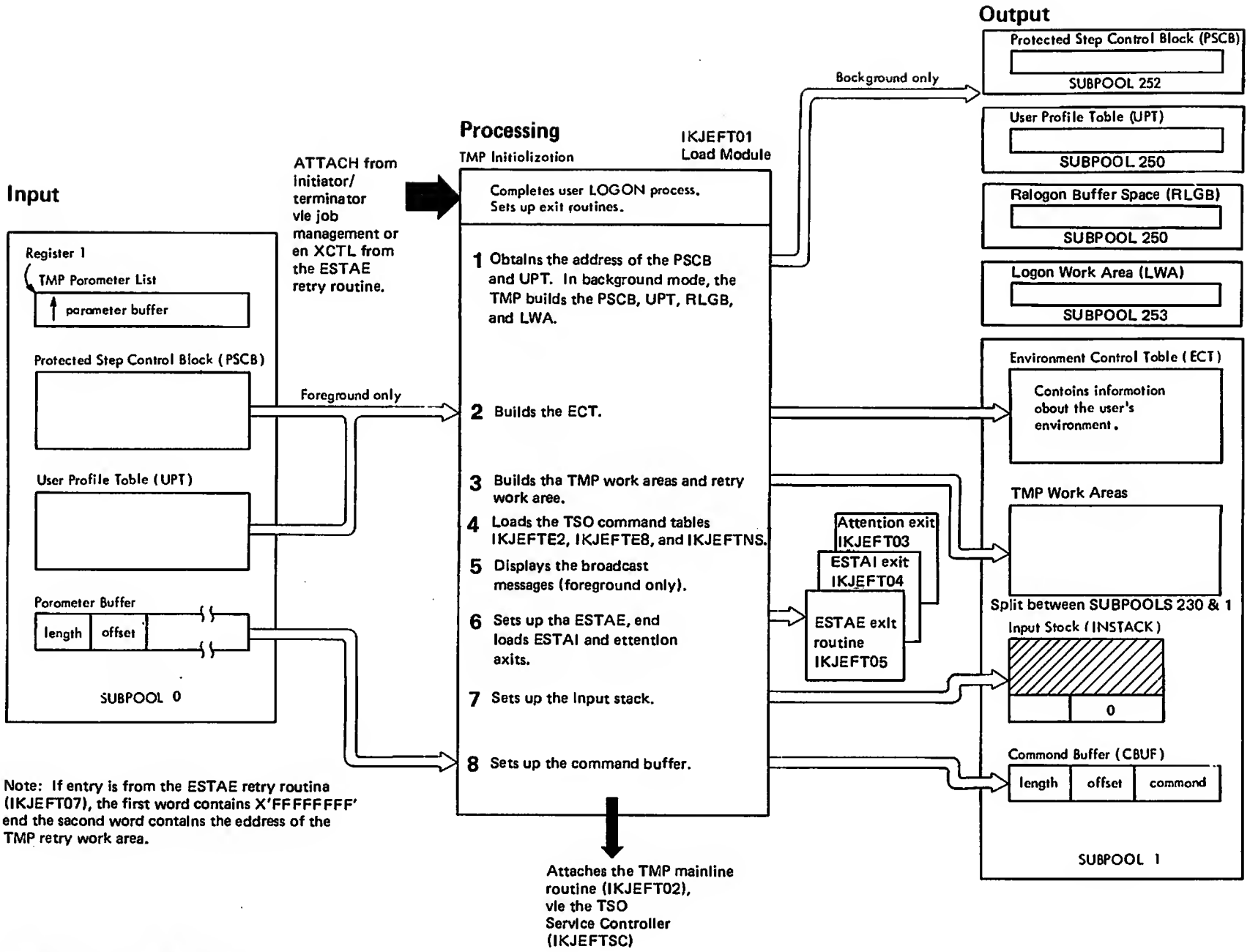


Diagram 2. TMP Initialization (Part 1 of 2)

Key	Description	Routine	Label
1	The TMP initialization routine uses an EXTRACT macro instruction to obtain the address of the protected step control block (PSCB). The PSCB contains the address of the user profile table (UPT). (In background mode, the TMP initialization routine builds the PSCB, the UPT, the RLGB, and the LWA.)	Initialization	IKJEFT01
2	The TMP initialization routine constructs the environment control table (ECT) from information the PSCB and UPT contain.		
3	The TMP initialization routine builds the following major internal work areas:		
	1. TMPWRKA1-contains control information for normal processing.		TMPWRKA1
	2. TMPWRKA2-contains control information for IKJEFT02.		TMPWRKA2
	3. TMPWA2-contains control information for IKJEFT02.		TMPWA2
	4. TMP3-contains control information for parallel TMP processing.		TMP3
	5. TSP-contains control information for LAR processing of TMP I/O.		TSP
	6. TIB-is a communication block between parallel TMPs.		TIB
4	The TMP initialization routine calls IKJEFTP1 to either load the command tables IKJEFTE2, IKJEFTE8 and IKJEFTNS, and put their addresses in the LWA fields (LWATE2, LWATE8, and LWATNS) if the tables are found in the LPA or in LINKLIB, or use the tables built in PARMLIB.	Initialization	IKJEFTP1
5	The TMP displays the broadcast messages if processing in foreground mode.		
6	The TMP initialization routine sets up the ESTAE exit by loading the ESTAE exit routine and issuing an ESTAE macro instruction. (See Diagram 3.)		
	IKJEFT01 loads the attention routine (IKJEFT03) in the foreground and puts the address in TMPWRKA1. IKJEFTSC establishes the attention routine (IKJATTN).	IKJEFTSC	
	The TMP initialization routine loads the ESTAI exit routine and the TIME command processor.		IKJEFT01
	If an error occurs while loading the TIME command processor, a retry is attempted and processing continues without loading TIME.	Initialization	IKJEFT01
7	The TMP initialization routine initializes the first element on the input stack to describe the terminal as the current source of input. (In the background, the TMP initialization routine uses the DATASET operand of the STACK macro to describe the SYSIN data set as the current source of input.)		
8	The TMP initialization routine sets up the command buffer (CBUF) and initializes it with the value from the PARM field of the first EXEC statement in the user logon procedure.		

Diagram 2. TMP Initialization (Part 2 of 2)

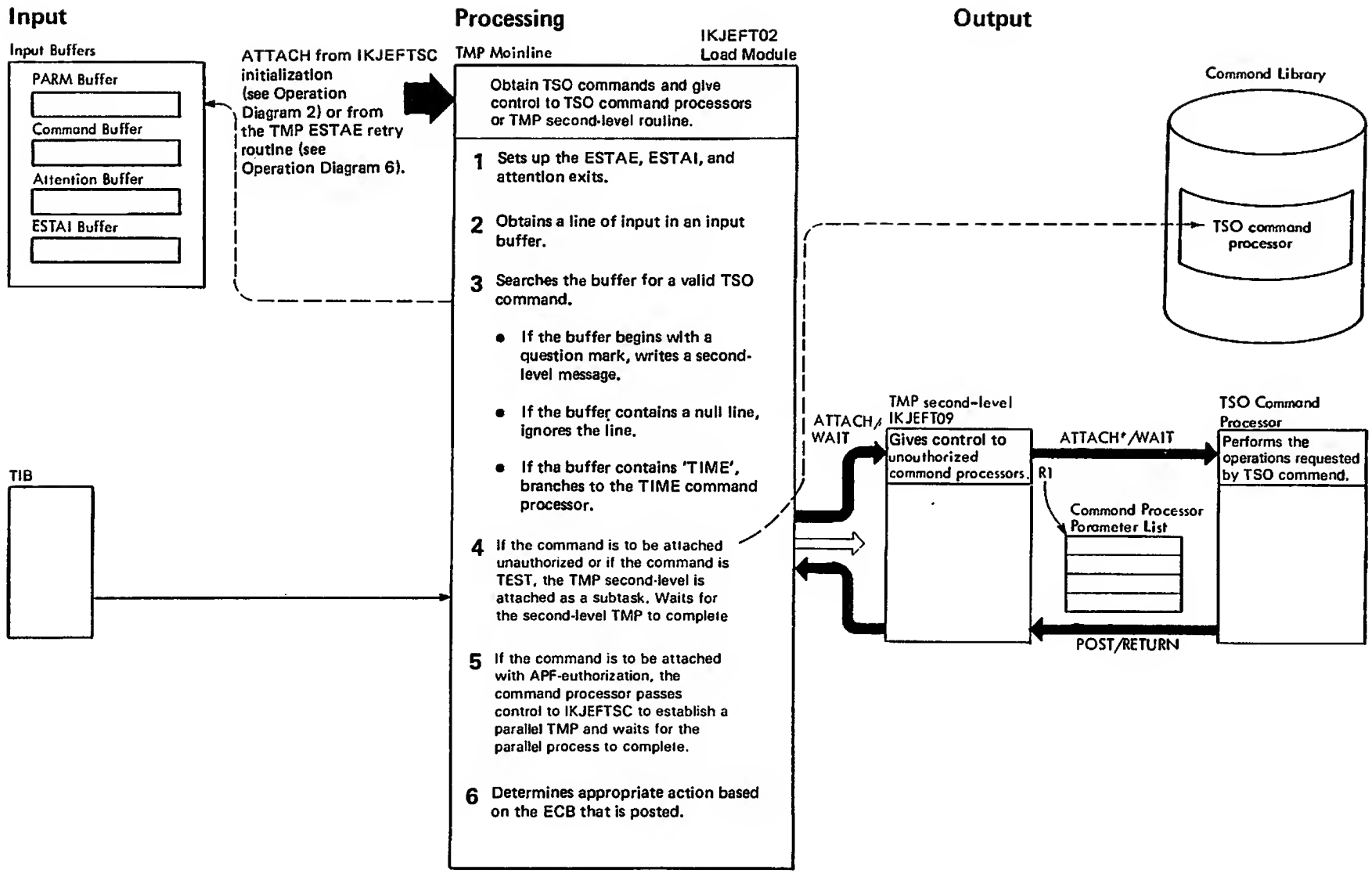
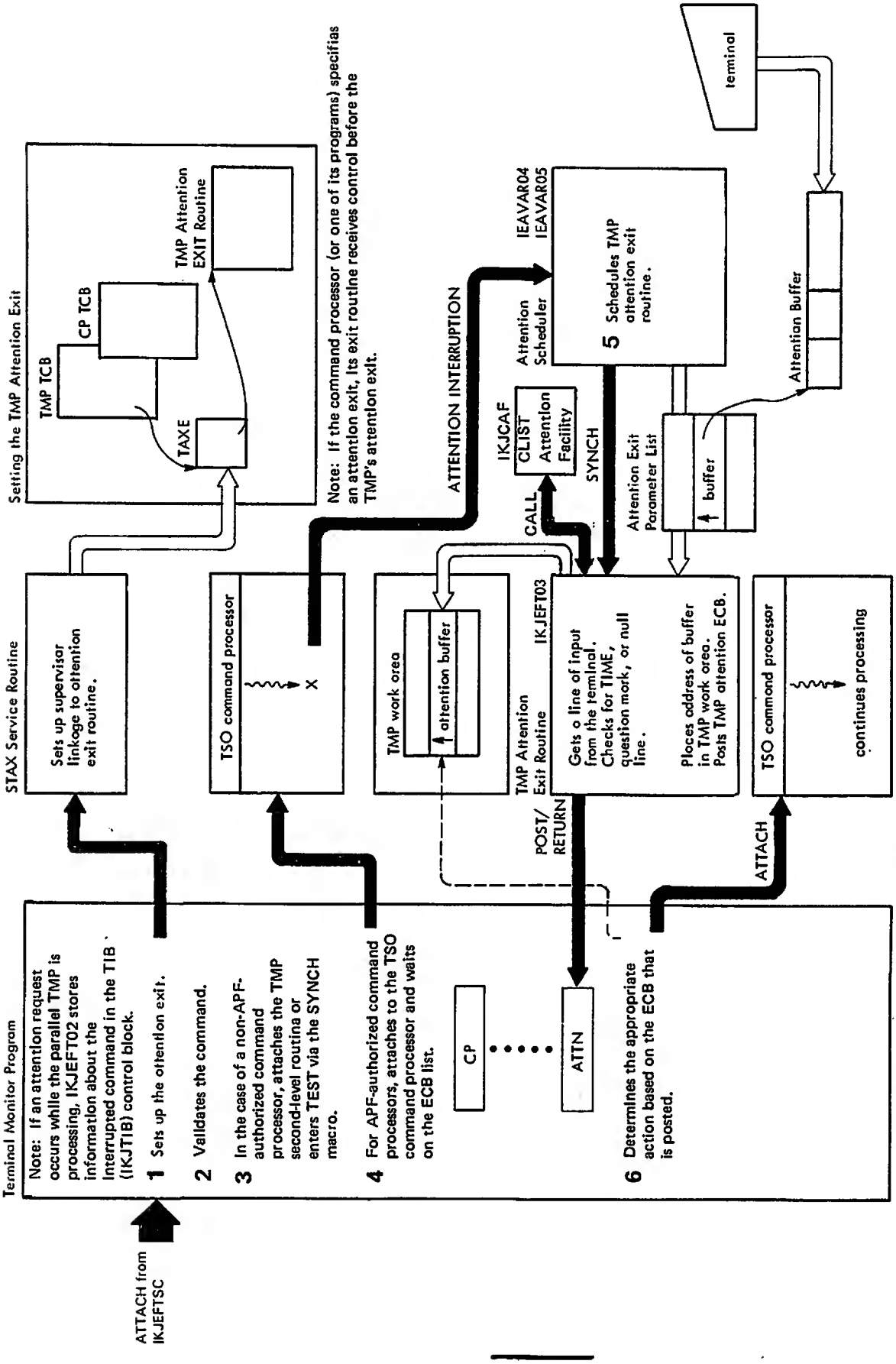


Diagram 3. Handling TSO Commands with the Primary TMP (Part 1 of 2)

Key	Description	Routine	Label	Diagram
1	The TMP mainline routine sets up the ESTAE exit by issuing an ESTAE macro instruction. The TMP mainline routine issues a STAX macro to establish the attention exit (IKJEFT03). The TMP mainline routine sets up the ESTAI exit by including the ESTAI operand on the ATTACH macro instruction when giving control to a command processor.			
2	The TMP mainline routine obtains a TSO command from one of four input buffers, depending upon the situation:			3
	• At entry from the initiator/terminator, the TMP mainline routine obtains a command from the PARM buffer.	TMP Mainline	IKJEFT02	2
	• When handling an attention request, the TMP obtains a command from the attention buffer pointed to by the CMDWAIT field of the TMP work area.	TMP Mainline	ATTNPOST	5
	• When handling an ESTAI request, the TMP obtains a command from the command buffer.		STAIPOST	6
	• When a command processor completes processing, the TMP obtains a command from the command buffer.		GETCMD	3
	• When a return code from the parallel TMP indicates that an attention interrupt or an abend occurred, the primary TMP obtains the next command pointed to by the TIBCMDBF field of the TIB control block.			
3	The TMP mainline routine uses the command SCAN to search the buffer for a valid command name. In most cases the TMP mainline routine attaches a subtask (see Steps 4 and 5). Following are the four cases when the TMP processes the contents of the buffer without attaching a subtask:		SCAN	3
	Buffer Contains — **Action Taken** question mark — Writes chained second-level messages to the terminal. null line — Ignores the line. invalid command — Writes an error message to the terminal. TIME — Branches to the TIME command processor.			
	After the requested processing has been completed, the TMP prompts the terminal for another command.			
4	In the case of a non-APF-authorized command processor, the TMP mainline routine attaches the TMP second-level to either attach the processor as a subtask or to enter TEST via the SYNCH macro. The TMP mainline routine waits for the TMP second-level routine to complete. The dispatcher gives control to the TMP's subtask(s) and allows the subtask to execute.			
5	In the case of an APF-authorized program or command processor, IKJEFTSC passes control to a parallel TMP (IKJEFT02) and waits for it to complete. The TMP mainline routine waits for the command processors to complete. The dispatcher gives control to the TMP's subtask(s) and allows the subtask to execute.			
6	After an ECB is posted and the TMP regains control from the dispatcher, the TMP determines why it got control and takes the appropriate action:		LIST	3
	ECB Posted — **Action Taken** CP — Obtains another command (see Step 2). ATTN — Obtains another command (see Step 2). ESTAI — Obtains another command (see Step 2). TIBRECB — Obtains another command (see Step 2). LOGOFF — Performs cleanup operations before returning to the initiator/terminator.			

Diagram 3. Handling TSO Commands with the Primary TMP (Part 2 of 2)

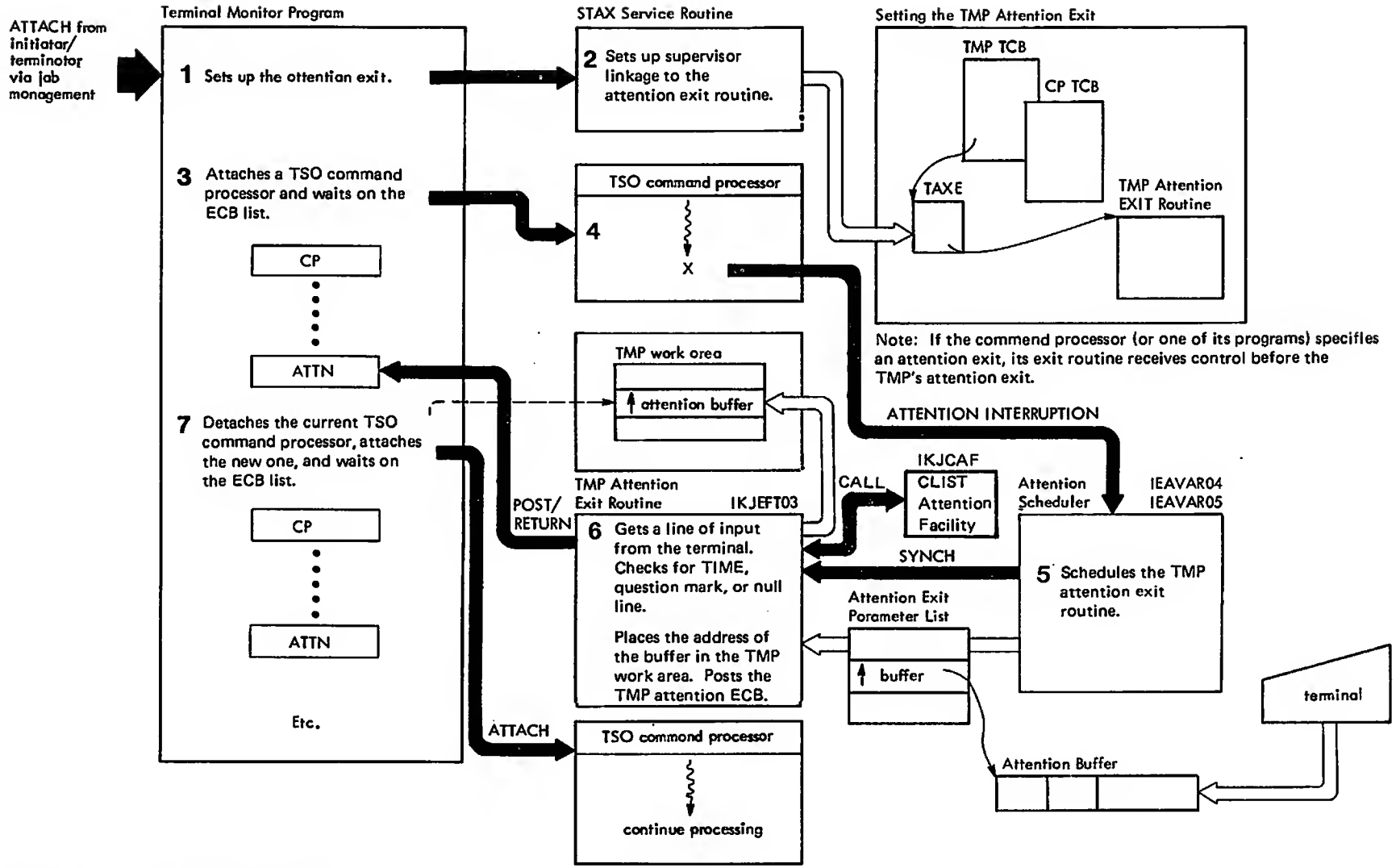


Note: The above diagram describes attention processing for both the APF-authorized and non-APF-authorized command processors. However, for the purpose of this diagram, the TMP mainline and TMP second-leval are shown as a single task.

Diagram 4. Handling TSO Commands with a Parallel TMP (Part 1 of 2)

Key Description	Routine	Label	Diagram
1 The TMP mainline routine sets up the attention exit (IKJEFT03).			
2 The TMP mainline routine obtains a TSO command from one of three input buffers, depending upon the situation:		IKJEFTSC	3
• At entry from the TMP, obtains a command from the TIB block.			2
• When handling an attention request, the TIB is used to pass information about the attention interrupt.		ATTNPOST	5
• When handling an ESTAI request, the TMP obtains a command from the command buffer.		STAIPOST GETCMD	6 3
3 The TMP mainline routine uses the command SCAN to search the buffer for a valid command name. In most cases, the TMP initialization routine attaches a subtask (see Steps 4 and 5). Following are the four cases when the TMP processes the contents of the buffer without attaching a subtask:		SCAN	3
Buffer Contains — **Action Taken** question mark — Writes chained second-level messages to the terminal. null line — Ignores the line. invalid command — Writes an error message to the terminal. TIME — Branches to the TIME command processor.			
4 In the case of a non-APF-authorized command processor, the TMP mainline routine attaches the TMP second-level routine to attach the processor as a subtask or to enter TEST via the SYNCH macro. The routine waits for the completion of the TMP second-level routine. The dispatcher gives control to the TMP's subtask(s) and allows the subtask to execute.			
5 The TMP mainline routine attempts to attach any APF-authorized command processor directly; however, if the processor is not found, the TMP assumes the command processor is an EXEC and the buffer contains a valid member name. The routine waits for the completion of the command processor. The dispatcher gives control to the TMP's subtask(s) and allows the subtask to execute.		IMPLEXEC	3
6 After an ECB is posted and the TMP mainline routine regains control, the routine determines why it got control and takes the appropriate action:		LIST	3
ECB Posted — **Action Taken** CP — Obtains another command (see Step 2). ATTN — Obtains another command (see Step 2). ESTAI — Obtains another command (see Step 2). LOGOFF — Performs cleanup operations before returning to the initiator/terminator.			
The parallel TMP (IKJEFT02) puts the completion information in the TIB and returns control to IKJEFTSC.			

Diagram 4. Handling TSO Commands with a Parallel TMP (Part 2 of 2)



Note: The above diagram describes the attention processing for both the APF-authorized and non-APF-authorized commend processors. However, for the purpose of this diagram, the TMP mainline and TMP second-level are shown as a single task.

Diagram 5. Handling Attention Reqnests (Part 1 of 2)

Key	Description	Routine	Label	Diagram
1	The TMP initialization and mainline routines issue a STAX macro instruction to specify an attention exit routine.	TMP Initialization TMP Mainline	IKJEFT01 IKJEFT02	2 3
2	The STAX service routine sets up the necessary control blocks and returns control to the TMP.	STAX Service Routine	IKJEFTSC	4
3	The TMP initialization and mainline routines continue processing and obtaining commands, attaching command processors, and waiting on an ECB list. See Diagram 4 for more detail.	TMP Mainline TMP second-level	IKJEFT02 IKJEFT09	3 3
4	At some point during the processing of a command, the user presses the attention key.			
5	The attention scheduling routine in the region control task (RCT) gets control and schedules the TMP attention exit routine.	ATTN Scheduling Routine	IEAVAR04 IEAVAR05	4
6a	IKJEFT03 passes control to the CLIST attention facility (IKJCAF). IKJCAF determines if the attention exit is for a CLIST that has a CLIST attention exit. If so, IKJCAF issues a PUTGET to obtain the TSO command in the CLIST attention exit. Otherwise, control is passed back to IKJEFT03. IKJEFT03 checks the return code passed back from IKJCAF and issues a PUTGET with the READY mode message if this is a normal attention interrupt.	TMP Attention Exit	IKJEFT03	4
6b	The attention handling routine uses command scan to scan the attention buffer for a valid command. If a null line is entered, the command processor that was operating at the time of the attention resumes processing. The routine handles TIME and question mark requests directly. For other cases, the routine marks the interrupted task(s) as non-dispatchable by issuing the STATUS macro with the STOP operand. The routine then moves the new command into the command waiting field of the TMP work area and posts the ATTN ECB (IOPLECB).			
6c	If an attention interrupt occurs during parallel TMP processing that was initiated by the TSO service routine SVC (IGX00035), and the request was not to process a CLIST, the currently executing program or command processor terminates. The TIB (IKJTIB) control block contains the return code.			
7	When the TMP initialization and mainline routines regain control, they find that their ATTN ECB has been posted. If the interrupted command processor is APF-authorized, the TMP mainline routine detaches the processor. If the interrupted command processor is non-APF-authorized, the TMP mainline routine detaches the TMP second-level routine.	TMP Mainline	IKJEFT02	3

Diagram 5. Handling Attention Requests (Part 2 of 2)

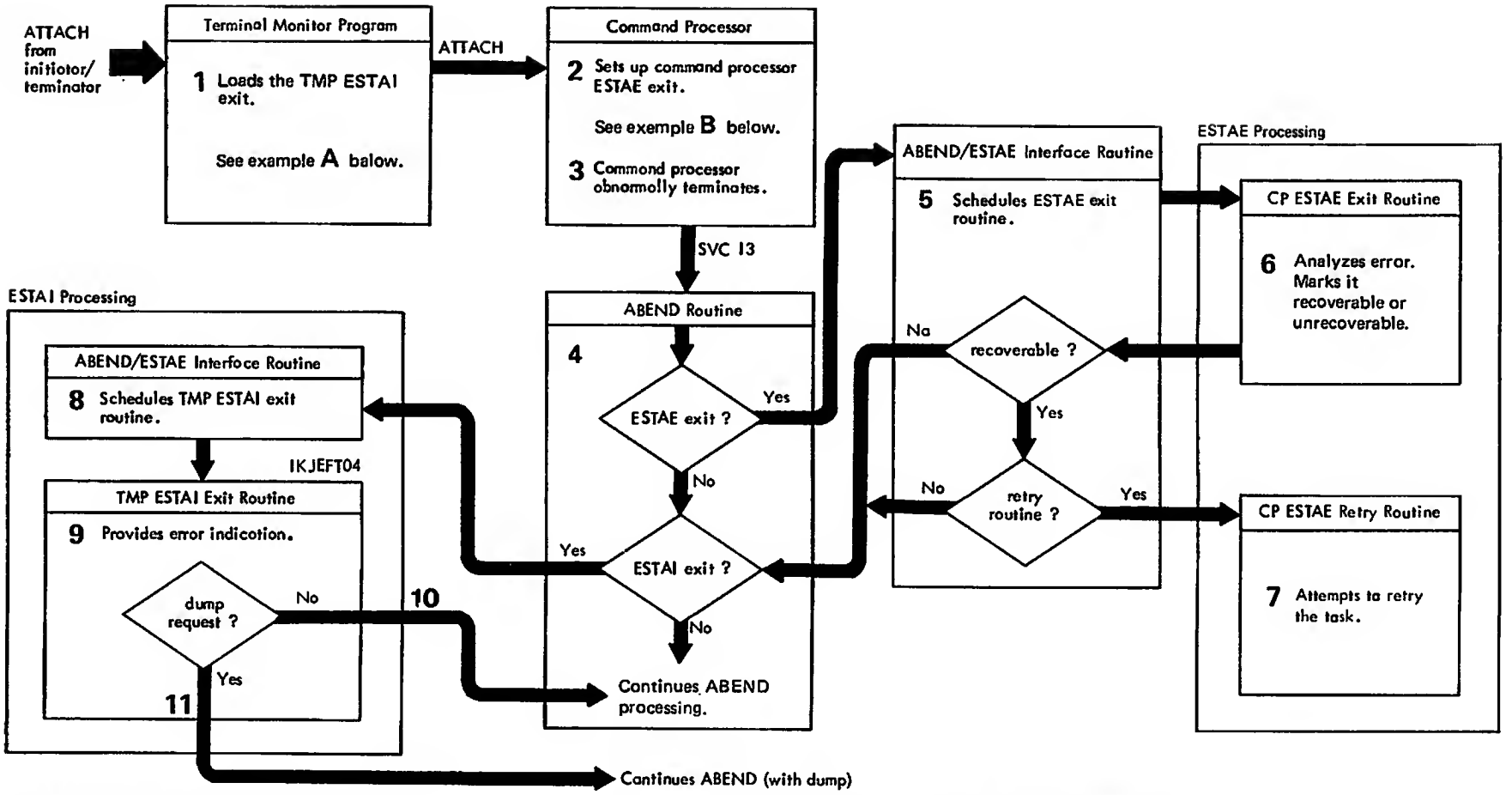


Diagram 6. Handling ESTAI Requests (Part 1 of 2)

Key Description	Routine	Label	Diagram
Note: The ABEND/ESTAE interface routine and the ABEND routine are both part of the recovery termioation manager (RTM).			
1 Duriog initializatioo, the TMP initialization routine loads the TMP ESTAI exit routine. Later, when the TMP mainlioe routioe gives control to a command processor, it issues an ATTACH macro iostructioo with ao ESTAI operand. The ESTAE service routine builds ao ESTAI cootrol block specifying the address of the ESTAI exit routine and chains it to the TCBNSTAE field of the command processor's TCB.	TMP Initialization TMP Mainline TMP second-level ESTAE Service Routine	IKJEFT0I IKJEFT02 IKJEFT09	2 3
2 The command processor, duriog its initialization process, may issue an ESTAE macro instruction specifyiog the address of an ESTAE exit routine and, possibly, an ESTAE retry routine.	Command Processor		
3 When an error in the command processor results io an ABEND, control passes to the ABEND routine.			
4 The ABEND routine finds ao ESTAE exit routine and passes control to the ABEND/ESTAE interface routine.			
5 The ABEND/ESTAE ioterface routine quiesces all active I/O, purges all ready I/O, attempts to estahlish a work area, aod schedules the command processor's ESTAE exit routine by issuing a SYNCH macro instruction.			
6 The command processor's ESTAE exit routine analyzes the error and marks it recoverable or unrecoverable.	Command Processor		
7 If the error is recoverable, the ABEND/ESTAE interface routioe attempts to pass control to the ESTAE retry routine. If the retry routine exists, it attempts to retry the failing task.	Commaod Processor ESTAE Retry		
8 If a successful CP ESTAE retry did not occur, the TMP ESTAI exit routine is scheduled using a SYNCH macro instructioo.	ABEND/ESTAE Interface		
9 The TMP ESTAI ECB is posted. If the TEST command processor had cootrol, TEST is reentered.	TMP ESTAI Exit	IKJEFT04	5
Otherwise, the user is prompted for a command.	TMP Mainline	IKJEFT02 STAIPOST	3
10 If a command other than TEST is entered, control is returned to the ABEND routine, or if TEST is entered, invokes the TEST processor.	ABEND Routine		
11 If a null lioe is eotered, control returns to the ABEND routine with a dump request. To obtaio a dump either the SYSABEND, SYSUDUMP, or SYSMDUMP DDNAME must be allocated.			
Note: The handliog of ESTAI requests for both APF and non-APF-authorized command processors are described above. However, for the purpose of this diagram the TMP mainline and TMP secood-level are showo as a siogle task.			

Diagram 6. Handling ESTAI Requests (Part 2 of 2)

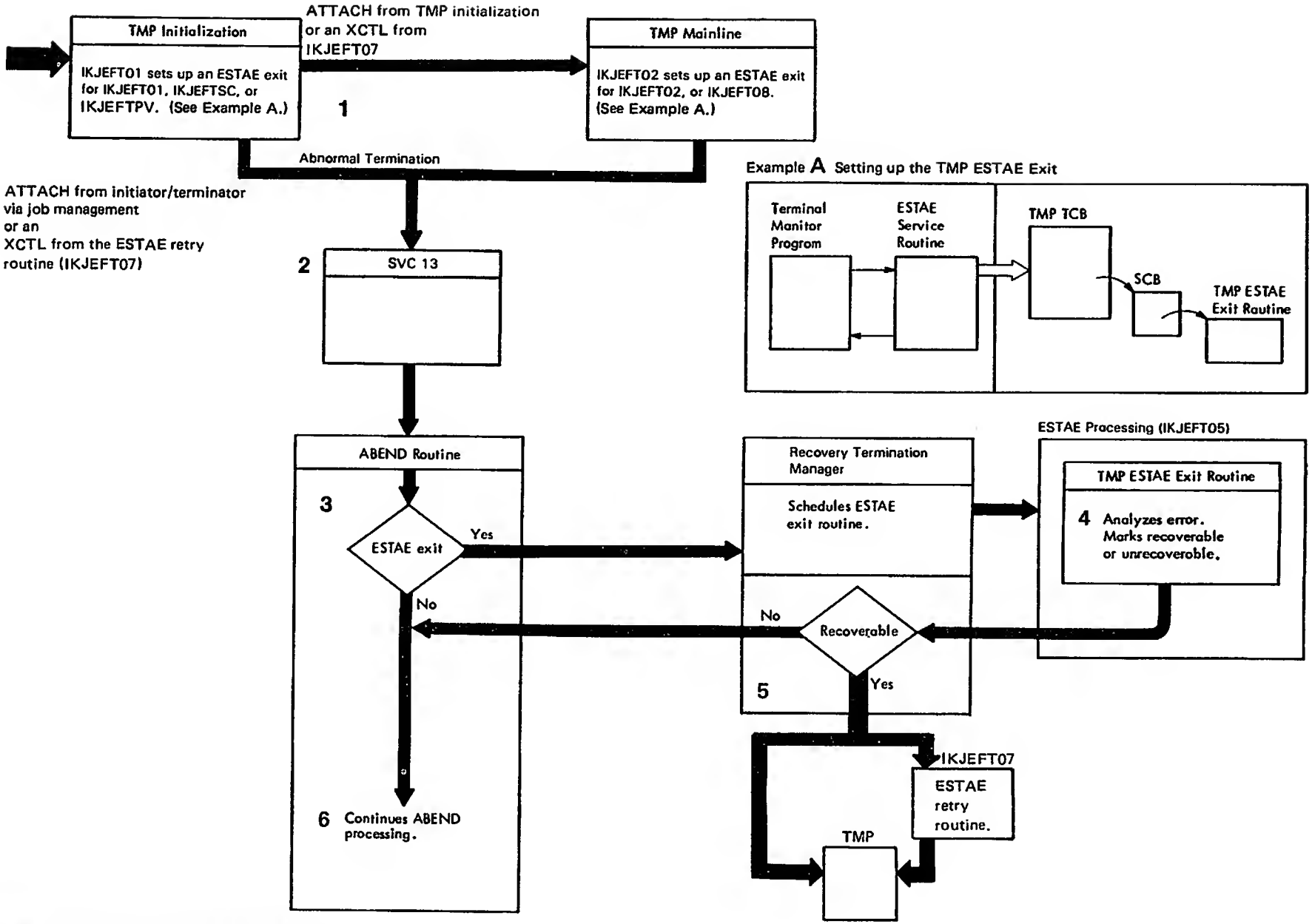


Diagram 7. Handling ESTAE Requests (Part 1 of 2)

Key	Description	Routine	Label	Diagram
1	The TMP initialization and mainline routines (both IKJEFT01 and IKJEFT02) issue an ESTAE macro instruction that sets up parameters for the ESTAE service routine. The ESTAE service routine builds an ESTAE control block (SCB) that contains the address of the TMP ESTAE exit routine.	TMP Initialization Mainline	IKJEFT01 IKJEFT02	2
2	When a recognizable error occurs in any TMP routine, an ABEND (SVC 13) is issued.	ABEND SVC		
3	The ABEND routine checks the TCBNSTAE field of the TCB for the abnormally terminating task and, finding that an ESTAE exit is specified, passes control to the recovery termination manager.	ABEND SVC		
4	The TMP ESTAE exit routine diagnoses the cause of the error and marks the error recoverable or not recoverable.	TMP ESTAE exit routine	IKJEFT05	6
5	If the error is recoverable, the TMP is re-initialized, if necessary, and processing resumes.	TMP ESTAE Retry Routine	IKJEFT07 IKJPVRET IKJTTERM IKJRECPT IKJTWAIT IKJBETSC	6
6	If the error is not recoverable, control is returned to ABEND for abnormal termination processing.	ABEND SVC		

Diagram 7. Handling ESTAE Requests (Part 2 of 2)

Program Organization

This section describes the organization of the TMP. It contains information about the hierarchy of the load modules, the assembly modules, and the control sections that constitute the program. The module operation information briefly describes the processing operations that occur within each TMP module.

Program Hierarchy

The TMP contains the following load modules: IKJEFT01, IKJEFT02, IKJEFT04, IKJEFT07, IKJEFTSL, IKJTABLS, and IKJEFT09. The TMP initialization routines (IKJEFT01 and IKJEFTP1) complete the logon process by setting up tables and control blocks, and by setting up the ESTAE, ESTAI, and attention exit routines. The command tables may be obtained from a PARMLIB data set member. For a description of the PARMLIB data set, see *TSO Extensions Customization*.

The TMP mainline routine (IKJEFT02) obtains TSO commands, attaches the second-level TMP (IKJEFT09), gives control to TSO command processors, and monitors their execution.

The TMP consists of the following object modules:

IKJEFT01 - TMP initialization routine.

IKJEFT02 - TMP mainline routine.

IKJEFT03 - TMP attention exit routine (contained in load module IKJEFT02).

IKJEFT04 - TMP ESTAI exit routine (contained in load module IKJEFT04).

IKJEFT05 - TMP ESTAE exit routine (contained in load module IKJEFT04).

IKJEFT06 - TMP message module (contained in the load modules IKJEFT01, IKJEFT02, IKJEFT04, and IKJEFT07).

IKJEFT07 - TMP ESTAE retry routine.

IKJEFT08 - TMP CALL processor routine (contained in load module IKJEFT02).

IKJEFT09 - TMP second-level routine (contained in load module IKJEFT02).

IKJEFT01 - Service routine work area initialization routine (contained in load modules IKJEFT01 and IKJEFT04).

IKJEFTAP - Attach programs via the TSO service facility table (contained in load module IKJTABLS).

IKJEFTE2 - Attach commands with RSAPF table (contained in load module IKJTABLS).

IKJEFTE8 - Attach programs via CALL with RSAPF table (contained in load module IKJTABLS).

IKJEFTNS - Non-supported command name table (contained in load module IKJTABLS).

IKJEFTP1 - Contains initialization routines (performs processing for IKJEFT01).

IKJEFTP2 - TMP mainline stage 1 procedures (contained in load module IKJEFT02).

IKJEFTPV - TSO service facility parameter verification (contained in load module IKJEFT01).

IKJEFTSC - TSO Service Controller attaches any TMP mainline routine (IKJEFT02) (contained in load module IKJEFT01).

IKJEFTSL - TMP linkage assist routine for data management I/O.

Control Flow

Figure 2-1 and Figure 2-2 show the control flow among the object modules.

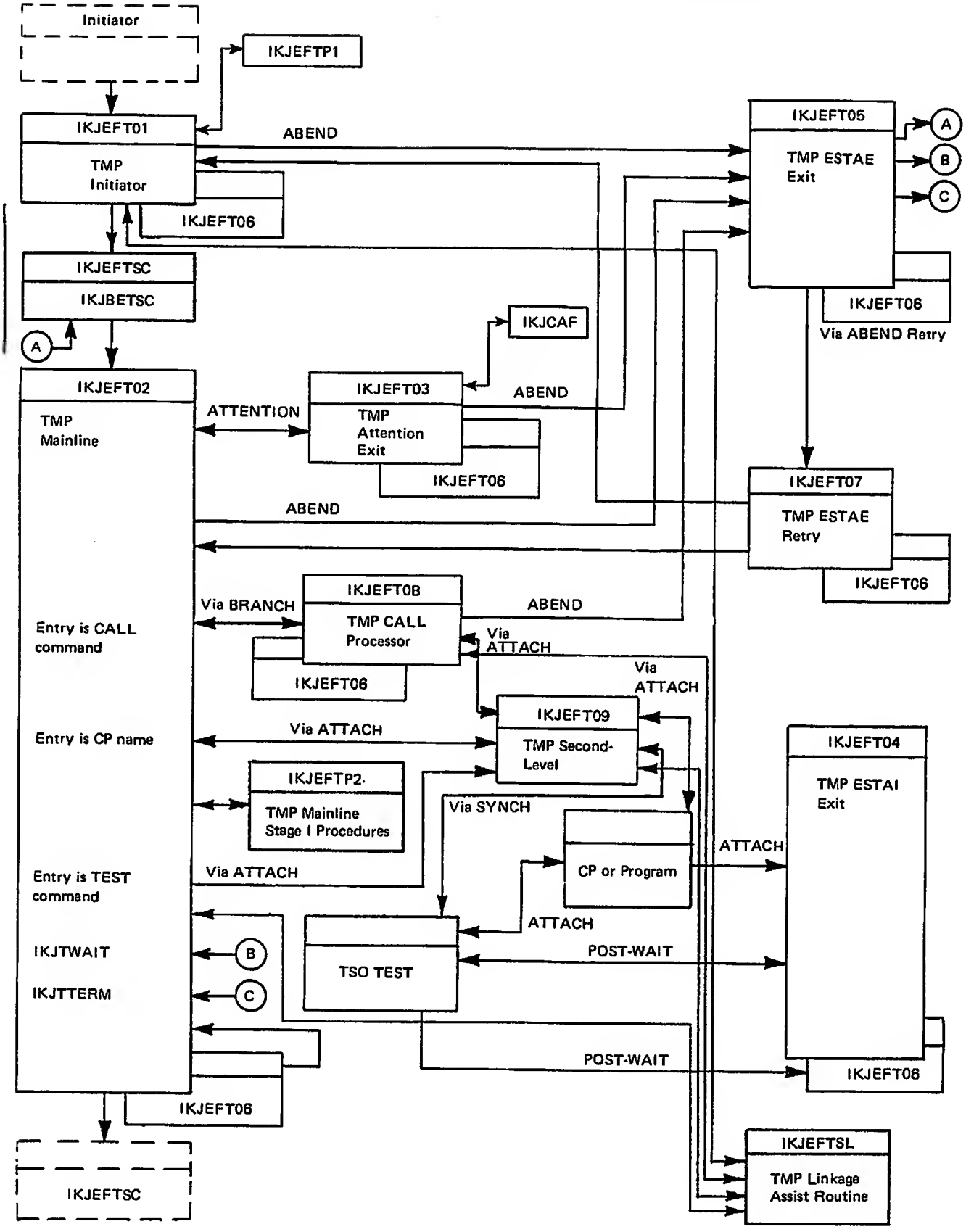


Figure 2-1. TMP Control Flow for a Non-APF-authorized Command Processor or Program

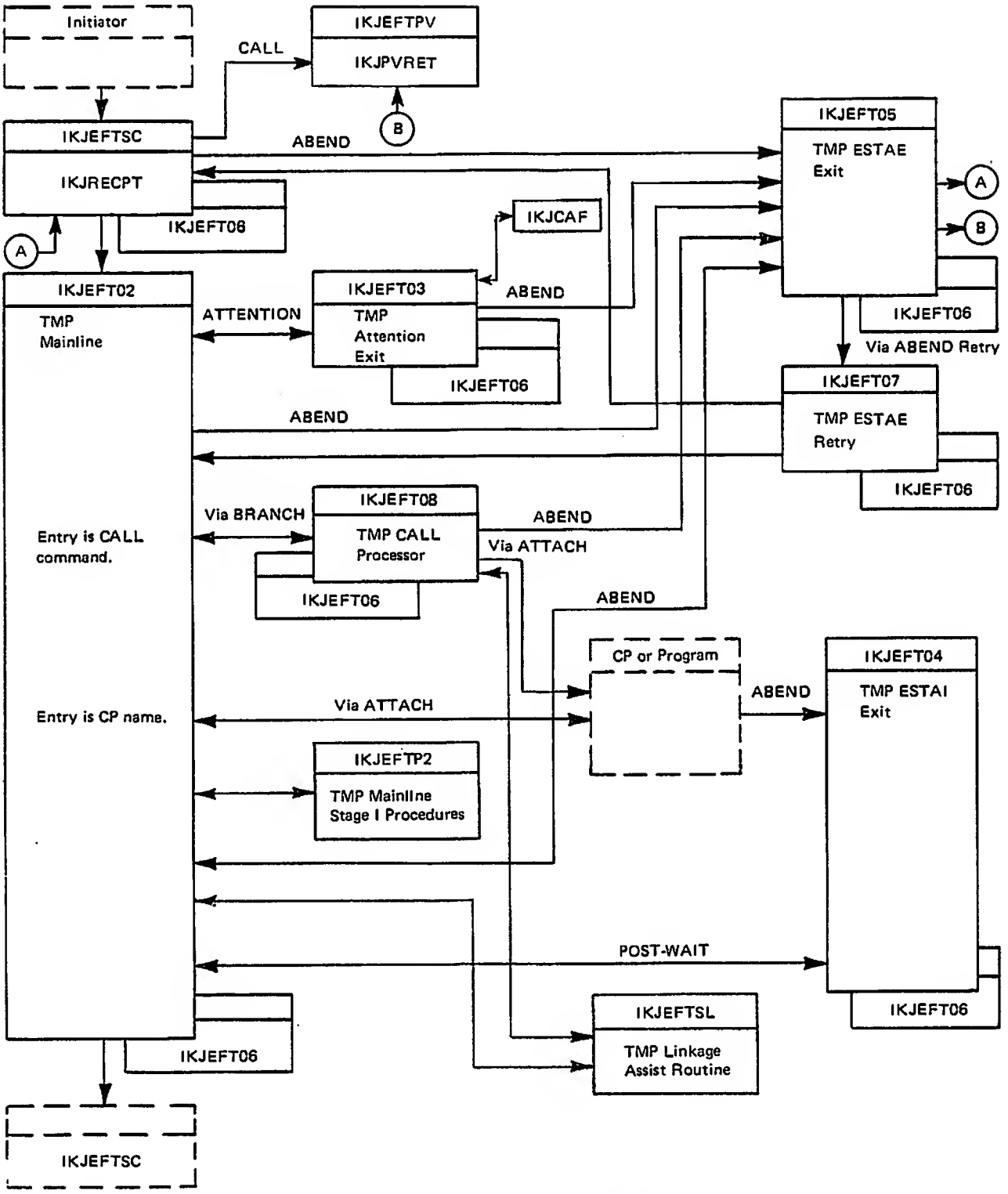


Figure 2-2. Parallel TMP Control Flow - APF Authorized Command Processor or Program

Module Operation

The following descriptions explain the tasks each executable TMP module performs:

IKJEFT01 -- TMP Initialization Routine

IKJEFT01 obtains and initializes tables and work areas that are used during the TSO session, enables the ESTAE exit routine, and attaches LISTBC (IKJEES73). On the first call to IKJEFT01, it builds the first input stack element and initializes the command buffer from the LOGON parameter field, then calls the TSO service controller (IKJEFTSC).

IKJEFT01 performs the following operations:

- Gains addressability to the UPT by issuing the EXTRACT macro to get the PSCB, which contains the UPT address. (In background mode, IKJEFT01 builds the PSCB and the UPT.) It uses this information to construct the ECT.
- Builds four major internal work areas:
  - The TMP work area, TMPWRKA1, which contains parameter lists and control information for normal operation of the TMP.
  - The TMP retry work area, TMPWA2, which contains information needed by the TMPESTAE retry routine.
  - The TMPWRKA2, a protected work area that contains information needed by the TMP mainline to indicate what processing the mainline needs to perform.
  - The TMP3 work area, used for creating a parallel side.
- Builds images of TMPWRKA1 and TMPWRKA2 to be used for parallel processing.
- Obtains storage for, and initializes, the TSP. The TSP is chained off of TMPWRKA2.
- Sets up the ESTAE exit by issuing the ESTAE macro.
- If operating in foreground, loads the attention exit.
- Sets up the ESTAI exit by loading the ESTAI exit routine and including the ESTAI operand on the ATTACH macro. The ATTACH is performed when the TMP gives control to the command processor.
- Loads the command processor.
- If this is the first call to IKJEFT01, it attaches the LISTBC LOGON processor (IKJEES73).
- Calls IKJEFTP1 to store the addresses of the TSO command tables (IKJEFTE2, IKJEFTE8, IKJEFTNS, and IKJEFTAP) into the LOGON work area (LWA).
- Initializes the first element on the input stack.
  In foreground mode, sets the terminal to be the current source of input.
  In background mode , uses SYSIN as the data set operand of the STACK macro to describe SYSIN as the current source of input.
- Sets up the command buffer (CBUF) and initializes it with the value from the parameter field of the EXEC statement in the user LOGON parameter.
- Calls IKJEFTSC to continue TMP mainline processing.

- Frees all storage that was GETMAINed by IKJEFT01 and regains control from IKJEFTSC.
- Passes control to the caller of IKJEFT01.

IKJEFT02 -- TMP Mainline Routine

IKJEFT02 performs initialization, obtains a command name, gives control to the appropriate command processor or to the TMP second-level routine, and waits for it to complete before obtaining another command.

IKJEFT02 performs the following operations:

- Establishes a TMP exit routine and a TMP attention routine.
- Obtains first command (may be NULL) from IKJEFT01; obtains subsequent commands using PUTGET.
- Checks command name for validity using command scan.
  - If invalid, issues diagnostic message.
  - If null, obtains another command.
  - If question mark, sends any second-level messages queued by the last command processor using PUTLINE.
  - If TIME, obtains running time for the terminal session by branching to the TIME command processor.
  - If TEST, attaches the TMP second-level (module IKJEFT09) to allow the user to test a program by entering the TEST command processor via the SYNCH macro.
  - If in background mode, checks command-name against names in non-supported name table (IKJEFTNS for background acceptability). If unacceptable, suppresses command processor invocation and issues a diagnostic message.
- If the command processor has been attached unauthorized, detaches the TMP second-level routine and its subtask. Otherwise, detaches the previous command processor.
- If a program or command processor is APF (authorized program facility) authorized, and running under a parallel TMP, IKJEFT02 attaches it with the RSAPF keyword parameter.
- If a program or command processor is APF-authorized and not running under a parallel TMP, IKJEFT02 posts IKJEFTSC to cause a parallel TMP to be initiated. IKJEFTSC then waits for the parallel TMP to complete processing before it continues.
- If a program or command processor within a CLIST is APF-authorized and a parallel TMP is processing the CLIST through the TSO service facility, IKJEFT02 posts IKEFTSC to initiate another parallel TMP. IKJEFTSC then waits for the third parallel TMP to complete processing before it continues.
- Attaches the TMP second-level (module IKJEFT09) to allow the attaching of the appropriate non-APF-authorized command processor.
- Waits on an ECB list. The operating system dispatcher gives control to the appropriate command processor.

- After returning from the dispatcher following a post, does one of the following:
  - Detaches a normally completed APF-authorized command processor and returns to the primary TMP. (End of task ECB posted.)
  - TMP second-level detaches non-APF-authorized command processor and returns control.

    TMP mainline detaches the TMP second-level and gets another command processor. (End of task ECB posted.)
  - Abnormally detaches the TMP second-level if the command processor has been attached non-APF-authorized; otherwise, abnormally detaches the previous command processor, and gets another command. (Attention ECB or ESTAI ECB posted.)
  - Issues a LOGOFF command.
  - In background mode, issues STACK macro to delete input/output data sets and issues FREEMAIN for the PSCB, UPT, LWA, and RLGB space.

IKJEFT03 -- TMP Attention Exit Routine

IKJEFT03 is the attention handling routine for the TMP. It processes attention interrupts that occur while executing under the TMP.

IKJEFT03 calls the CLIST attention facility (IKJCAF) to determine if there is a CLIST that contains an attention exit to process. IKJEFT03 bases further processing of the attention interrupt on the return code from the CLIST attention facility. The return codes and the associated processing are as follows:

- 0 - The CLIST attention facility handled a CLIST attention exit and IKJEFT03 continues processing by scanning the input buffer that is passed back from IKJCAF.
  - If the command is TIME, IKJEFT03 will SYNCH to the TIME command processor and pass back a null line to IKJEFT02.
  - If the command is found to be invalid, IKJEFT03 prints out a message and passes back a null line.
  - If the command is found to be a null line, control passes back to the executing program at the point of interruption.

  Any other valid command is passed back to IKJEFT02 to be processed.
- 4 - MVS system product version 2.2.0 is not installed on the system and IKJEFT03 handles the CLIST attention exit as follows:
  - IKJEFT03 issues a PUTGET to process the CLIST attention exit. Control remains in IKJEFT03's PUTGET until a TSO command is retrieved. The following are special cases:
    - When a null line is retrieved instead of a TSO command:

      If the user entered an attention interrupt while a TSO command was executing, IKJEFT03 ignores the null line and processing continues at the point of interruption.

      If the user entered an attention interrupt while a CLIST statement was processing, IKJEFT03 passes back a null line because IKJEFT02 is looking for a command to be passed back. IKJEFT02 recognizes the

null line, ignores it, and issues a PUTGET to continue processing the CLIST.

– When TIME or an invalid command is found in the CLIST attention exit:

IKJEFT03 processes the request or the error and passes back a null line to IKJEFT02. IKJEFT02 processes the CLIST at the next command after the attention.

Note: If MVS system product version 2.2.0 is not installed on the system and an attention interrupt occurs while processing a CLIST attention exit, control is passed to the highest level attention exit (IKJATTN in IKJEFTSC). IKJATTN posts an attention ECB; IKJEFT02 checks this ECB and SYNCHes to IKJEFT03.

- 8 - The attention interrupt was not for a CLIST that contains a CLIST attention exit and IKJEFT03 processes a normal attention interrupt based on the environment established prior to the attention interrupt.

  If an SVC-initiated parallel TMP environment exists, an attention return code is placed in the TIB (TIBRC) and processing terminates for the interrupted command processor. Otherwise, a PUTGET is issued to obtain the next command.

  If either an IKJEFT02-initiated environment or a non-parallel environment exists at the time of the attention interrupt, the user may enter a line of input consisting of:

  – A null line (by pressing Enter):
  This causes the attention interrupt to be ignored and processing continues at the point of interruption.

  – TIME or question mark:
  The READY mode message is reissued after the TIME request or question mark is processed.

  – A valid command:
  If a non-parallel environment exists, this command is placed on the TMP wait list and IKJEFT02 is posted. Otherwise, an IKJEFT02-initiated parallel environment exists and the command is placed in the TIBCMDBF, then IKJEFT02 is posted.

  – An invalid command:
  An invalid command causes the user to be prompted for a valid command or a null line.

  If TEST is active, IKJEFT09 must be restarted by issuing the STATUS macro instruction to allow TEST to return to IKJEFT09. IKJEFT09 then returns to IKJEFT02, which continues normal attention processing.

- 16 or greater - The CLIST attention facility terminated abnormally and IKJEFT03 issues a user ABEND 304 to let the IKJEFT05 ESTAE routine clean up and start the TMP processing once again.

IKJEFT04 -- TMP ESTAI Exit Routine

IKJEFT04 informs the terminal user that a task is terminating abnormally. IKJEFT04 performs the following operations:

- Issues a diagnostic message.
- Posts the TMP ESTAI ECB.
- Waits on an ESTAI exit ECB.

If recovery is possible (TEST had control), IKJEFT04 marks the task recoverable and returns to ABEND. If recovery is impossible (TMP had control), IKJEFT04 returns to ABEND.

IKJEFT05 -- TMP ESTAE Exit Routine

IKJEFT05 intercepts abends in TMP modules and determines whether to retry or continue with termination.

Retry is possible if the recovery termination manager (RTM) provides a system diagnostic work area (SDWA) and IKJEFT05 gathers the following information to be placed in the SDWA:

- The name of the module that had control when the ABEND occurred, based on footprint bits contained in the variable TMPWRKA2.
- The environment when the ABEND occurred, namely one of the following:
  - A non-parallel environment (that is, one TMP mainline task IKJEFT02 exists) without a TMP interface block (TIB)
  - A non-parallel environment with a TIB
  - A parallel environment (that is, a second TMP mainline task IKJEFT02 executing concurrently on behalf of authorized commands or programs executing under the TSO service facility).
- Whether previous ABENDs occurred in this module, determined by examining recursion bits in the variable TMPWKA2.
- Whether the TMP is executing in the background or foreground.

If recovery is not possible, IKJEFT05 returns to the RTM and indicates that termination should continue.

If recovery *is* possible in a non-parallel environment without a TIB, IKJEFT05:

- Notifies the user of the ABEND condition.
- Takes an SVC dump.
- Writes a software record to the SYS1.LOGREC data set.
- Loads either the ESTAE retry routine, IKJEFT07, or selects a retry point in the failing module.
- Returns to RTM, which in turn passes control either to IKJEFT07 or the retry point in the failing module. If the module fails again, IKJEFT05 tries to reinitialize the TMP. IKJEFT05 does not take additional SVC dumps when it tries to reinitialize the TMP.

If recovery is possible in a parallel environment or a non-parallel environment with a TIB, IKJEFT05:

- Notifies the user of the ABEND condition.
- Takes an SVC dump.
- Writes a software record to the SYS1.LOGREC data set.
- Selects a retry point in the failing module.
- Returns to RTM, which in turn passes control to the retry point in the failing module. If the module fails again, IKJEFT05 tries to reinitialize the TMP. IKJEFT05 does not take additional SVC dumps when it tries to reinitialize the TMP.

IKJEFT07 -- TMP ESTAE Entry Routine

IKJEFT07 processing depends upon whether a TMP restart has been attempted for this task:

- If no TMP restart has been attempted, control passes to the TMP mainline routine (IKJEFT02) for a restart.
- If a TMP restart has been attempted, control passes to the TMP initialization routine (IKJEFT01) for re-initialization of IKJEFT02, 03, 04, 05, and 25.

IKJEFT08 -- Call Function

IKJEFT08 calls the command processor that is to be passed control.

IKJEFT09 -- TMP Second-Level

IKJEFT09 is attached by the TMP mainline or CALL command processor. IKJEFT09 gives control to the appropriate non-APF-authorized command processor and waits for its completion. It is detached by the TMP mainline when the current command processor completes processing or terminates.

IKJEFT09 performs the following operations:

- Upon receiving control from either the TMP mainline or CALL command processor, the TMP second-level does one of the following:
  - Attaches a command processor.
  - Attaches a called program.
  - Enters TEST via a SYNCH macro.
- Waits for the ECBs to be posted for completion or abnormal termination of the command processor.
- Detaches the command processor or enters TEST via the SYNCH macro.
- Closes and frees the chain of DCBs that may exist if the RUN subcommand of TEST was executed for a load module. The pointer to the DCB chain is in the TPLETDCB field of the TPLE control block. The TPLTPL field of the TPL control block points to the TPLE.

  Note: The DCB of the TASKLIB data set is not closed when TEST ends because the problem program can continue to run. If the DCB for the TASKLIB data set were closed, the problem program would abend when the TASKLIB was referenced.
- Is detached by the TMP mainline routine.

IKJEFTP1 -- TMP Initialization Subroutines

IKJEFTP1 contains a series of external procedures used by the TMP initialization routine, IKJEFT01.

IKJEFTP1 performs the following operations:

- TMP background initializations:
  - Gets space (in the background mode) for the UPT, PSCB, RLGB, and LWA.
  - Initializes blocks with information from either the UADS or RACF (if RACF is available), otherwise it uses the default values.
- Command/program/exit tables processing:
  - A BLDL is done to locate the tables in STEPLIB or in the LNKLST. If found in either place, then a copy of each table found will be used by issuing a LOAD command. If a table is not found, the table built through the PARMLIB process is used and will be pointed to by the LWA. If PARMLIB was not used to build a table, the copy in LPA will be pointed to by the LWA.

IKJEFTP2 -- TMP Mainline Stage 1 Procedures

IKJEFTP2 contains a series of external routines for IKJEFT02.

IKJEFTSC -- TSO Service Controller

Attaches the mainline TMP (IKJEFT02). The TSO service controller gives control to the mainline TMP IKJEFT02, and waits until it receives a request to:

- LOGOFF
- Initialize a parallel TMP.

In the case of LOGOFF, IKJEFTSC cleans up all work areas, frees storage, and returns control to IKJEFT01. When a request for a parallel TMP is received, the TSO service controller:

- Creates the work areas (TMPWRKA2,TMPWRKA1, TMPWA, TPLE, and TSP) and initializes them.
- Sets the invoker of the service non-dispatchable.
- Quiesces all I/O for the requestor.
- Calls IKJEFTPV to validity check the parameters supplied by the caller of the TSO service facility.
- Creates a parallel task structure via an ATTACH macro with the ECB parameter specified.
- Waits for the parallel task structure to complete.

- When the authorized ECB is posted, indicating that the parallel TMP has completed, IKJEFTSC:
  - Detaches the parallel task structure.
  - Posts the requestor thru the ECB in the TIB control block.
  - Restores I/O for the requestor.
  - Resets the requestor's task dispatchable.
  - Waits for another request.

IKJEFTPV -- TSO Service Facility Parameter Verification

IKJEFTPV checks the validity of the parameters passed by the caller of the TSO service routine (IKJEFTSR). It verifies that the parameters are in storage and available to the caller. When IKJEFTPV is invoked it,:

1. Verifies that the number of parameters is valid (either 6 or 7).
2. Verifies the caller's authority to read the parameters.
3. Verifies that all flags have values and that the reserved fields are set to zero.
4. Verifies that the TSO service facility (TSF) function buffer is not longer than 32K minus 5 bytes. The TSF function buffer (a field in the parameter list) contains the name of the program or command to be invoked and the commands' operands.
5. Verifies the caller's authority to write in the return fields.
6. If a seventh parameter is present, verifies the caller's authority to read in the program parameters.
7. Passes validation information back to the TSO Service Controller (IKJEFTSC).

Note: For additional information concerning the TSO service routine or its parameter list, see *TSO Extensions Programming Guide.*

IKJEFTSL -- TMP Linkage Assist Routine (TMP LAR)

IKJEFTSL assists programs executing above 16 megabytes in virtual storage in using services that reside below 16 megabytes. The TMP LAR does the following:

- Performs standard linkage of save areas.
- Switches to 24-bit addressing mode.
- Issues a macro instruction to do one of the following:
  - Open a data set.
  - Close a data set.
  - Build a data set.
  - Find a name in a data set.
  - Read a record and check to see if I/O has been completed.
- If an I/O error occurred and IKJEFT01 was the caller, takes an I/O error (SYNAD) exit.
- Switches to 31-bit addressing mode.
- Restores the caller's registers.

Data Areas

IKJEFTAP, IKJEFTE2, IKJEFTE8, and IKJEFTNS are CSECTs in IKJTABLS that reside below 16 megabytes in virtual storage.

Note: Each of these tables can be built through TSO PARMLIB processing. It is not necessary to assemble and link the CSECTs into IKJTABLS.

IKJEFTAP -- Attach Programs via the TSO Service Facility Table

Table of names of programs to be attached by the TSO service facility with the RSAPF option to allow APF-authorized programs to run (for a description, see *TSO Extensions Customization*).

IKJEFTE2 -- Attach Commands with RSAPF Table

A table of command names to be attached with the RSAPF option to allow APF-authorized programs to run. (See *TSO Extensions Customization* for a description.)

IKJEFTE8 -- Attach Programs via CALL with RSAPF Table

A table of program names to be attached by CALL with the RSAPF option to allow APF-authorized programs to run. (See *TSO Extensions Customization* for a description.)

IKJEFTNS -- Non-Supported Command Name Table

A table of valid command names that are not supported in the background. Any such name is rejected with a diagnostic message.

Directory

This table contains information that helps you find the appropriate program description or assembly listing.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
IKJEFT0I	TMP Initialization	IKJEFT0I	IKJEFT01	IKJEFT01	Builds tables and work areas, sets up exit routines.
IKJEFT02	TMP Mainline	IKJEFT02	IKJEFT02	IKJEFT02	Gets commands, and supervises their execution.
IKJEFT03	TMP ATTN Exit	IKJEFT02	IKJEFT03	IKJEFT03	Handles ATTN request directed to the TMP.
IKJEFT04	TMP ESTAI Exit	IKJEFT04	IKJEFT04	IKJEFT04	Intercepts abnormally terminating command processors or program tasks.
IKJEFT05	TMP ESTAE Exit	IKJEFT04	IKJEFT05	IKJEFT05	Intercepts abnormally ending TMP or TEST command processor.
IKJEFT06	TMP Messages	IKJEFT01 IKJEFT02 IKJEFT04 IKJEFT07	IKJEFT06	IKJEFT06	Contains TMP messages.
IKJEFT07	TMP ESTAE Retry Routine	IKJEFT07	IKJEFT07	IKJEFT07	Deletes IKJEFT02, 03, 04, 05 and 25.
IKJEFT08	CALL Function	IKJEFT02	IKJEFT08	IKJEFT08	CALL command processor.
IKJEFT09	TMP Second-Level	IKJEFT02	IKJEFT09	IKJEFT09	Attaches and detaches non-APF-authorized command processors.
IKJEFT0I	SRWA Initialization Routine	IKJEFT0I IKJEFT04	IKJEFT0I	IKJEFT01	Initializes service routine work areas.
IKJEFTAP	APF Program Table (via the TSO service facility)	IKJTABLS	IKJEFTAP	IKJEFTAP	Table of programs to be attached by the TSO service facility with APF authorization
IKJEFTE2	APF Command Table	IKJTABLS	IKJEFTE2	IKJEFTE2	Table of commands to be attached with APF authorization.
IKJEFTE8	APF Program Table	IKJTABLS	IKJEFTE8	IKJEFTE8	Table of programs to be attached by CALL with APF authorization.
IKJEFTNS	Commands not supported (NS Table)	IKJTABLS	IKJEFTNS	IKJEFTNS	Table of command names not supported in background.
IKJEFTP1	TMP Init Procedures	IKJEFT01	IKJEFTP1	IKJEFTP1	Performs procedures for IKJEFT01.
IKJEFTP2	TMP Mainline Stage I Procedures	IKJEFT02	IKJEFTP2	IKJEFTP2	Performs procedures for IKJEFT02.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
IKJEFTPV	TSO Service Facility Parameter Verification	IKJEFT01	IKJEFTPV	IKJEFTPV	Verifies the validity of its parameters passed by the caller of IKJEFTSR.
IKJEFTSC	TMP Service Controller	IKJEFT01	IKJEFTSC	IKJEFTSC	Attaches the mainline TMP (IKJEFT02) or a parallel TMP.
IKJEFTSL	TMP Linkage Assist Routine	IKJEFTSL	IKJEFTSL	IKJEFTSL	Issues data management macros below 16 megabytes in virtual storage.
IKJEFTSR	TSO Service Routine	IKJEFTSR	IKJEFTSR	IKJEFTSR	Interface for TSO service facility (TSOLNK).

Diagnostic Aids

This section contains a summary of ABEND and return codes and their meanings.

Routine	Decimal ABEND Code	Meaning
IKJEFT01	102	STACK error.
	104	Background PSCB-UPT build error.
	105	PUTLINE error.
	106	Insufficient main storage to execute.
	207	STACK error.
	Note:	These are error exit (user ABEND) codes used by IKJEFT01 when going to IKJEFT05.
IKJEFT02	103	STAX error.
	201	ATTACH error.
	202	BLDL error.
	203	DAIR error.
	204	PUTGET error.
	205	PUTLINE error.
	206	Command Scan error.
	207	STACK DELETE = ALL error.
	208	IDENTIFY error.
	209	Error in parallel TMP.
	Note:	These are error exit (user ABEND) codes used by IKJEFT02 when going to IKJEFT05.
IKJEFT03	301	PUTGET error.
	302	PUTLINE error.
	303	Command Scan error.
	304	CLIST attention facility error.
	Note:	These are error exit (user ABEND) codes used by IKJEFT03 when going to IKJEFT05.
IKJEFT06	None	
IKJEFT07	701	STACK error.
	702	LOAD error.
IKJEFT08	205	PUTLINE error.
IKJEFT09	201	ATTACH error.
IKJEFTP2	103	STAX error.
	207	STACK DELETE = ALL error.
	209	Error in parallel TMP.
	217	STACK INQUIRE = ERROR error.
	227	STACK DELETE = BARRIER error.
IKJEFTSC	103	STAX error.
	106	Purge not successful.
IKJEFTSL	75	Invalid function type passed.

Routine	Decimal Return Code	Meaning
IKJEFTSL	00	Processing successful.
	04	I/O error.
IKJEFT04	00	Unrecoverable error.
	04	Recoverable error.
IKJEFT05	00	TMP restart is not to be attempted.
	04	TMP restart is to be attempted.

Chapter 3. Terminal I/O Service Routines

The terminal I/O service routines handle the terminal input/output operations required by the LOGON/LOGOFF scheduler, the TMP, the TSO/E command processors, and TSO/E problem programs.

There are four terminal I/O service routines:

- STACK
- GETLINE
- PUTLINE
- PUTGET

The terminal I/O service routines can be invoked directly with the LINK or LOAD/CALL macro instructions, or they can be invoked using the system macro instructions STACK, GETLINE, PUTLINE, and PUTGET.

When a program invokes an I/O service routine, the TSO/E service linkage assist routine (LAR) receives control. This routine resides below 16 megabytes in virtual storage and executes in the addressing mode in which it is invoked. It branches to the service routine using the BASSM instruction, which switches to 31-bit addressing mode. The service routine branches back to the TSO/E service LAR using the BSM instruction, which restores the original addressing mode. The TSO/E service LAR then restores the registers and returns control to the program.

The terminal I/O service routines reside above 16 megabytes in virtual storage and execute in 31-bit addressing mode, but can be invoked by programs executing in either 24-bit or 31-bit addressing mode. These routines can process input passed to them above or below 16 megabytes. All of the input data areas for these service routines can reside above or below 16 megabytes, except the list source descriptor (LSD), which is passed to the STACK routine.

For further information about the terminal I/O macro instructions, see *TSO Extensions Programming Services*.

For further information about the TSO/E service linkage assist routine, see "TSO/E Service Linkage Assist Routine" in this book.

Method of Operation

This section provides an overview of I/O service routine processing. It includes the following method of operations diagram:

- Diagram 8: Terminal I/O Service Routines (Overview) -- shows how the TMP, TSO/E command processors, and other TSO/E problem programs use the STACK, GETLINE, PUTLINE, and PUTGET terminal I/O service routines.

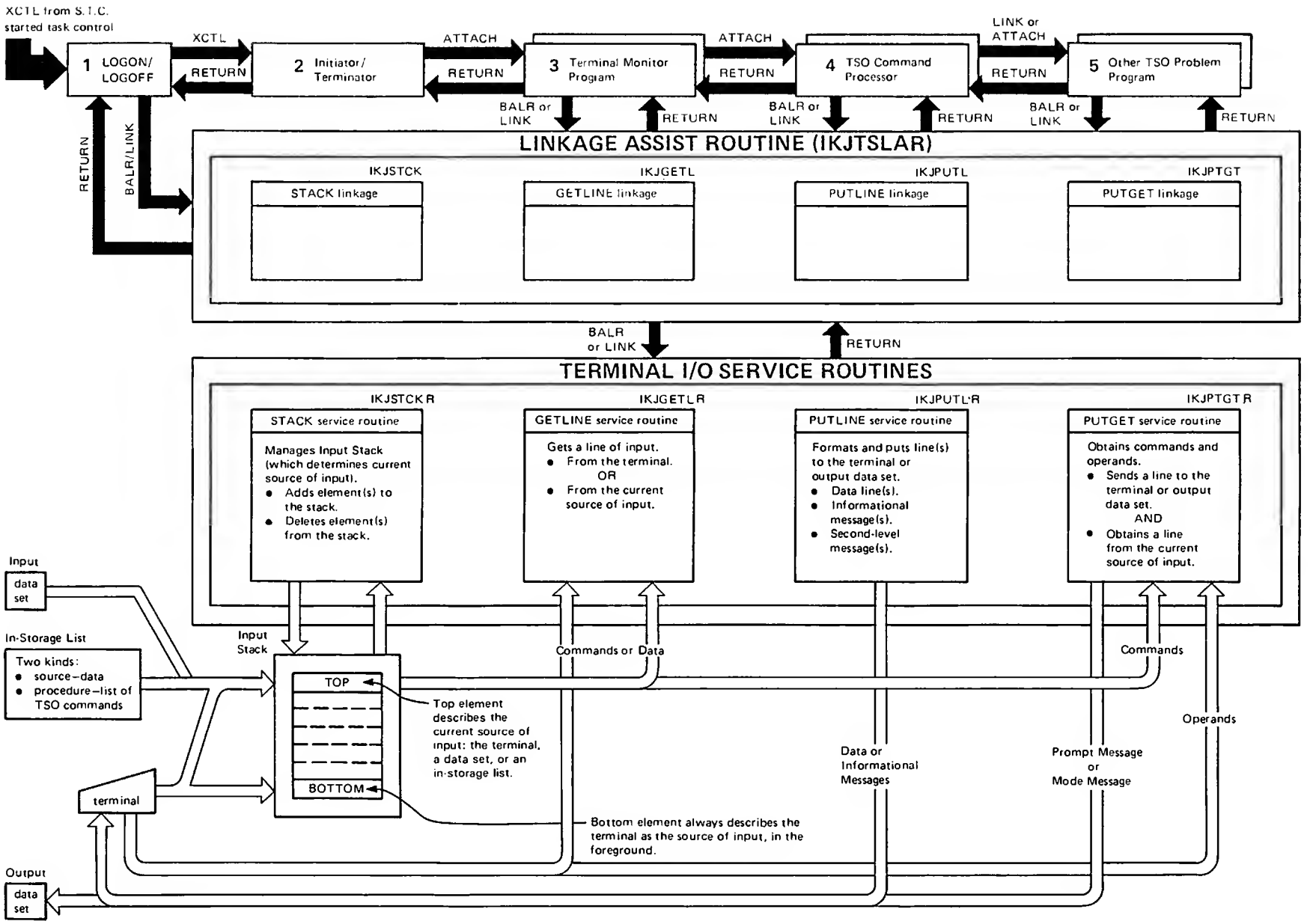


Diagram 8. Terminal I/O Service Routines (Overview) (Part 1 of 2)

Key	Description	Routine	
1	The LOGON/LOGOFF scheduler uses STACK to set the first (bottom) element to indicate the terminal as the current source of input, and uses other I/O service routines for terminal I/O during LOGON/LOGOFF prompting.	STACK GETLINE PUTLINE PUTGET	
2	The initiator/terminator attaches the TMP to process the terminal session.	TMP Initialization	
3	The TMP uses STACK to set the first (bottom) element to indicate either the terminal or a data set element in the background as the current source of input. This first (bottom) element is never removed.	STACK	
	The TMP uses PUTGET to obtain commands.	PUTGET	
4	The TSO/E command processors may use STACK to set up in-storage lists or data sets. There are three kinds of in-storage lists: source (data), procedure (TSO/E commands), and EXEC procedure (a procedure built by the EXEC command).	STACK	
	TSO command processors use PUTGET to obtain operands and subcommands. (Subcommand processors use PUTGET to obtain their operands.)	PUTGET	
	TSO command processors use GETLINE to obtain data and use PUTLINE to write first and second level messages or data to the terminal or the output data set.	GETLINE PUTLINE	
5	Other TSO problem programs may use any or all of the terminal I/O service routines if they share subpool 78 and use the proper parameter lists.	STACK GETLINE PUTLINE PUTGET	

Diagram 8. Terminal I/O Service Routines (Overview) (Part 2 of 2)

Program Organization

This section discusses the organization of the terminal I/O service routines: STACK, GETLINE, PUTLINE, and PUTGET.

Program Hierarchy

The following load modules make up the CLIST and terminal I/O service routines: IKJPTGTR IKJCT441, IKJCT442, IKJCTIOR, IKJCTTBL, IKJLDI00, and IKJLDI99.

Load module IKJPTGTR has the following control sections:

IKJCT431
IKJCT433
IKJCT434
IKJCT436
IKJCT437
IKJCT439
IKJCT440
IKJCT443
IKJCT444
IKJCT445
IKJCT446
IKJEFT30
IKJEFT35
IKJEFT40
IKJEFT45
IKJEFT52
IKJEFT53
IKJEFT54
IKJEFT55
IKJEFT56
IKJRBBMC

Load module IKJPTGTR has the following entry points:

IKJGETLR -- for GETLINE
IKJPTGTR -- for PUTGET
IKJPUTLR -- for PUTLINE
IKJSTCKR -- for STACK
IKJT441R -- for CLIST variable access

Load module IKJCT442 has the following control section:

IKJCT442

Load module IKJCTIOR has the following control section:

IKJCTIOR

Load module IKJCTTBL has the following control sections:

IKJCTTBL
IKJCT435

Load module IKJLDI00 has the following control sections, each of which is a LISTDSI CLIST statement module:

IKJLDI00
IKJLDI01
IKJLDI02
IKJLDI03
IKJLDI04
IKJLDI05

Load module IKJLDI99 has the following control section:

IKJLDI99 -- LISTDSI CLIST statement module

The following STACK functions are invoked as a result of STACK issuing SVC 109:

IGX00026
IGX00027

IGX00027 issues an abend code of X'66D' and the following associated reason codes:

1. Recovery could not be established.
2. Invalid stack address.
3. Stack is in use.
4. An invalid I/O service identifier was supplied by the caller.
5. A stack address is supplied in ECTIOWA but no stack table exists.
6. There is no stack. The STACK service routine should have been called but was not.
7. No stack table exists but STACK-TABLE is marked in use.
8. An ABEND 0C4 occurred when validating user pointers.

Diagnostic Aids

This section contains a summary of return codes and their meanings.

Routine	Return Code Hexadecimal	Meaning
STACK	00	Normal.
	04	Operation code could not be interpreted, or an invalid record was found in an in-storage list for an add request.
	08	Open fail on input that did not abnormally terminate.
	12	Open fail on output that did not abnormally terminate.
	16	Member specified was not found.
	18	A routine (attention or interrupt) exists.
	1C	A routine (attention or interrupt) does not exist.
	20	GETMAIN failure - not enough storage for data management control blocks (occurs only if MEMBER is specified).
	24	An INQUIRE function was specified and the requested CLIST routine (attention or error) was found on the STACK or SUBSTACK.
	28	An INQUIRE function was specified and the requested CLIST routine (attention or error) was not found on the STACK or SUBSTACK.
	2C	The INQUIRE=TYPE operand was specified and the topmost stack element is a barrier element.
	30	The INQUIRE=TYPE operand was specified and the topmost stack element is an input file name.
	34	The INQUIRE=TYPE operand was specified and the topmost stack element is an output file name.
	38	The INQUIRE=TYPE operand was specified and the topmost stack element has both an input file name and an output file name specified.
	3C	The INQUIRE=TYPE operand was specified and the topmost stack element is a TERMIN element.
	40	The INQUIRE=TYPE operand was specified and the topmost stack element is an unknown element.
	48	A barrier element is valid only with terminal elements.
PUTLINE	04	An attention interrupt occurred during PUTLINE processing.
	08	Nowait was specified and no line was returned.
	0C	The write function could not be interpreted because of faulty input parameters.
	10	A conditional GETMAIN was executed and there was insufficient space.
	14	A line drop has occurred.

Routine	Return Code Hexadecimal	Meaning
PUTGET	00	Normal.
	04	Input line not received from terminal.
	08	No input returned and/or no output line put out. (An attention interrupt occurred and the communications ECB was posted.)
	0C	1) Mode message was specified, input is not from the terminal, second-level messages are chained and user has specified no pause. 2) Prompt message was specified and: a)the user has specified no prompt, b)input is from a CLIST, or c)no CLIST prompt is available in background.
	10	Nowait was specified for the TPUT option and the output was not sent and input not received.
	14	Nowait was specified for the TGET option. Input was not received.
	18	Invalid parameters.
	1C	A conditional GETMAIN was issued and no space was available.
	20	A line drop occurred.
	28	A barrier element was on top of the STACK and SUBSTACK = YES was specified.
GETLINE	00	A line was obtained from the terminal.
	04	A line was obtained from the STACK or data set.
	08	An ECB was posted as a result of an attention interrupt.
	0C	Nowait was specified as a TGET option and a line could not be obtained from TGET.
	10	An EOD (end of data) condition from the STACK or data set.
	14	Invalid parameters were supplied to TGET or to the service routine.
	18	A conditional GETMAIN was executed and no space was available.
	1C	A line drop has occurred.
	20	A GETLINE, TERM request in background mode was unable to obtain a line from CLIST data prompt group.
	24	End of data received, continuation expected.
	28	A barrier element was on top of the STACK and SUBSTACK = YES was specified.

Routine	Return Code Hexadecimal	Meaning
IKJCT441	00	Normal.
	04	Variable returned should not be re-scanned.
	05	Variable returned requires evaluation.
	06	Variable returned requires evaluation.
	07	Variable returned requires evaluation.
	08	Variable returned requires evaluation.
	0C	Variable returned is a label.
	10	System variable - cannot be updated by the user.
	14	For locate - no variable returned. There are no more variables.
	18	Entry returned is a PROC name.
	20	GETMAIN/FREEMAIN failure.
	24	Symbol name or symbol value is too large or too small. (Entry is a procedure name.)
	28	Incorrect environment.
	2C	Invalid entry code.
	30	Duplicate symbol found.
	34	Undefined variable.
	38	Too many global variables.
	3C	Undefined global variable.
	40	SYSREF variable is not valid.
	44	Undefined SYSREF variable.
	48	Variable returned is a SYSREF variable.
	4C	Variable returned may be an installation built-in variable.

IKJCT441 is the CLIST symbolic substitution and variable access routine.

I/O Services Modules

The following is a list of the I/O services modules:

- IGX00027
- IKJCTIOR
- IKJEFT0I
- IKJEFT4A
- IKJEFT40
- IKJEFT45
- IKJEFT52
- IKJEFT53
- IKJEFT54
- IKJEFT55
- IKJEFT56
- IKJRBBMC

The following services are used by the I/O services modules:

- MVS services:
  - ABEND, ESTAE, FESTAE
  - GETMAIN, FREEMAIN, MODESET
  - SDUMP, SETRP, SETLOCK
  - VRADATA, TSOCALL, STAX
  - WTO, LINK, CPOOL.
- Data management services:
  - OPEN, CLOSE, READ
  - CHECK, RDJFCB, PUTX
  - PUT, GET, FIND
  - OBTAIN, LOCATE
  - SYNADRLS, SYNADF.
- TSO services:
  - TPUT, TGET, IKJFTPRT.

Chapter 4. Command Scan and Parse Service Routines

Command scan and parse search the command buffer for TSO/E commands and their parameters. In general, the TMP invokes command scan and the TSO/E command processors invoke parse. However, the TEST command processor and other TSO/E command processors that process subcommands also invoke command scan.

Command scan searches the command buffer for a question mark, command name, or null line. If syntax checking is requested, command scan checks the command name to be sure that it starts with an alphabetic character and contains no more than eight alphameric characters. If syntax checking is not requested, command scan assumes that the first alphabetic character in the buffer is the start of a command name. Command scan translates the command name to uppercase and updates the buffer offset to point to the first parameter or to the end of the buffer.

Parse searches the command buffer for command parameters, checks them for correct syntax, translates them to uppercase, and presents them to a user-supplied exit routine (if one is supplied) for validity checking. If a parameter is invalid, or if a required parameter is missing, parse can either prompt the terminal user to enter the parameter or supply a default value.

Command scan and parse accept double-byte character set (DBCS) strings in addition to EBCDIC character strings. The shift-out character, X'OE', indicates a change from EBCDIC to DBCS; the shift-in character, X'OF', indicates the reverse. Parse does not accept DBCS strings in prompting mode.

DBCS strings can appear within:

- Comments on TSO/E commands
- Quoted strings
- Self-delimiting strings
- Delimiter dependent positional parameters (parenthesized strings that are not being used as a variable, a keyword subfield, or a parenthesized expression).

An example of a DBCS string is:

```
aaa<d1d2d3>aaa
```

where:

aaa	Represents EBCDIC characters within a comment, delimiter dependent positional parameter, quoted string, or self-delimiting string.
<	Represents the shift-out character (X'0E')
d1d2d3	Represents three DBCS characters
>	Represents the shift-in character (X'0F')

As supplied with TSO/E, the command scan and parse service routines reside in SYS1.LPALIB and execute with the protection keys of their respective callers.

The command scan and parse service routines reside above 16 megabytes in virtual storage and execute in 31-bit addressing mode. These routines can process input above or below 16 megabytes in virtual storage, and can be invoked by programs executing in either 24-bit or 31-bit addressing mode. Those programs executing in 24-bit mode must use a linkage assist routine in order to use the service routines.

When a program invokes either the command scan or parse service routine, the TSO/E service linkage assist routine (LAR) receives control. This routine resides below 16 megabytes in virtual storage and executes in the addressing mode in which it was invoked. The LAR branches to the service routine using the BASSM instruction, which, if necessary, switches to 31-bit addressing mode. The service routine branches back to the TSO/E service LAR using the BSM instruction, which restores the original addressing mode. The TSO/E service LAR then restores the registers and returns control to the program.

Method of Operation

This section uses the following diagrams to describe the method of operation of the command scan and parse service routines:

- Diagram 9: Command Scan and Parse Service Routines (Overview) -- shows the basic functions command scan and parse perform.
- Diagram 10: Command Scan Service Routine -- shows how command scan searches the command buffer for TSO/E commands.
- Diagram 11: Parse Service Routine -- shows how parse searches the command buffer for TSO/E command parameters.
- Diagram 12: Parse Initialization -- shows how parse sets up the parameter descriptor list (PDL).
- Diagram 13: Searching the IKJPARSR Positional Parameters -- shows how parse searches the command buffer for IKJPARSR positional parameters.
- Diagram 14: Searching for IKJPARS2 Positional Parameters -- shows how parse searches the command buffer for IKJPARS2 positional parameters.
- Diagram 15: Searching for Keyword Parameters and Subfields -- shows how parse searches the command buffer for keyword parameters and subfields consisting of positional and/or keyword parameters.

Each diagram includes a cross-reference to help you find the appropriate program description or assembly listing.

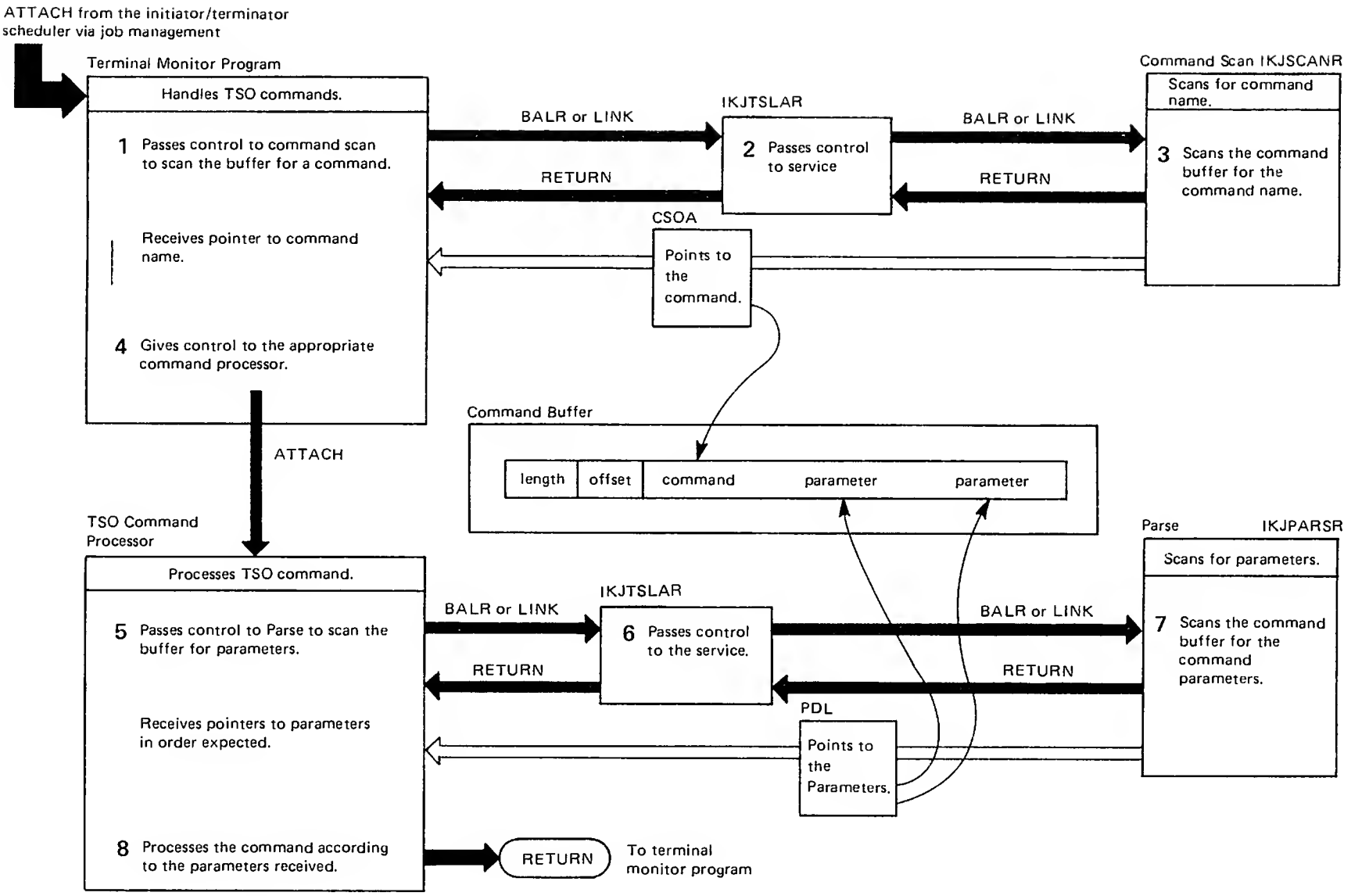
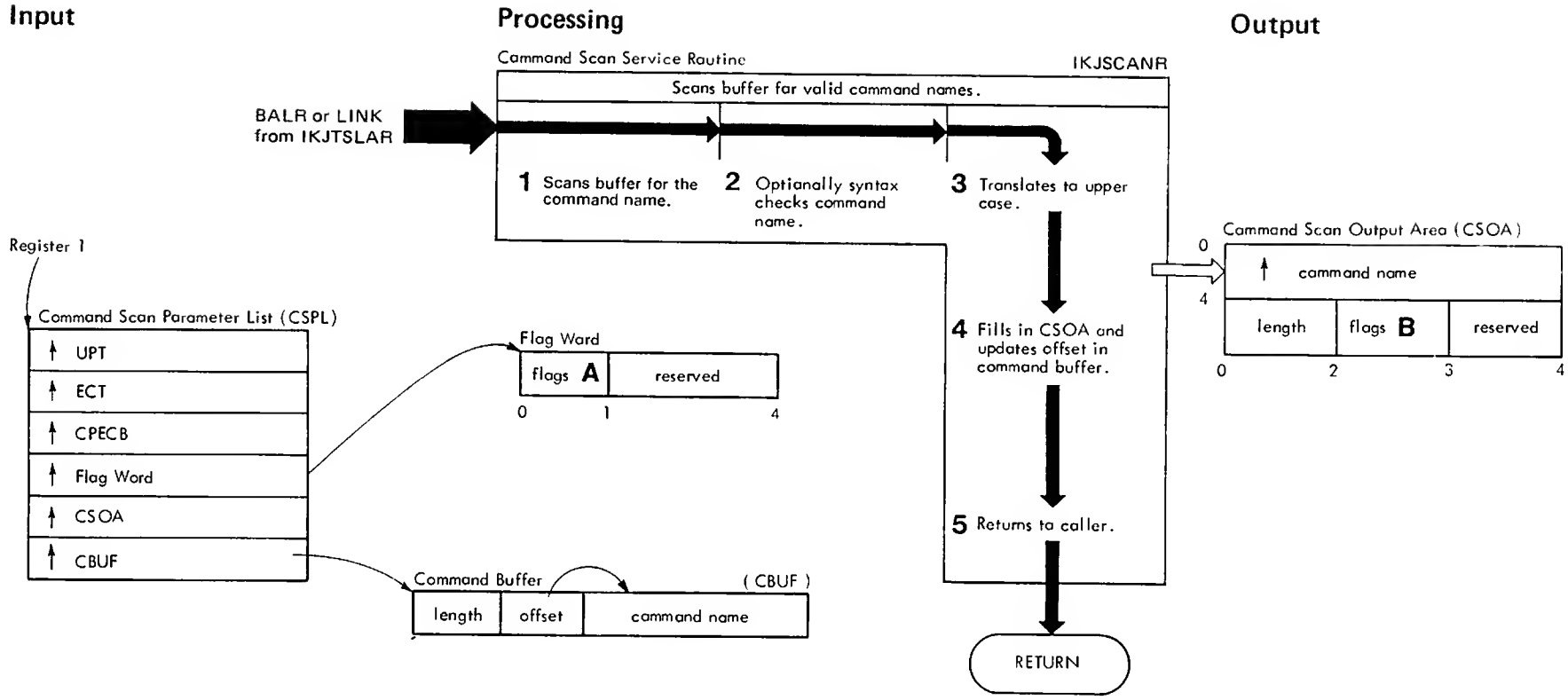


Diagram 9. Command Scan and Parse Service Routines (Overview) (Part 1 of 2)

Key	Description	Rontine	Label	Diagram
1	The terminal monitor program gets a line of input from the terminal. The TMP passes control to thc command scan service routine and passes it the address of the command buffer.	Terminal Monitor Program	IKJEFT02	3
2	If necessary, the linkage assist routine switches to 31-bit addressing mode and passes control to the command scan service routine.	TSO Service Linkage Assist Routine	IKJTSLAR	29
3	The command scan service routine scans the command buffer for a syntactically correct command name, updates the buffer offset field, and returns control to the TMP.	Command Scan Service Routine	IKJSCANR	17
4	The terminal monitor program receives the address of the correct command name and gives control to the appropriate TSO command processor.	Terminal Monitor Program	IKJEFT02	3
		Refer to Cbapter 2 of this book		
5	Thc TSO command processor passes control to the parse service routine and passes it the address of the command buffer and a parameter control list (PCL) that describes the parameters to be expected.	Refer to TSO command processors mauals		
6	If necessary, the linkage assist routine switches to 31-bit addressing mode and passes control to the parse service routine.	TSO Service Linkage Assist Routine	IKJTSLAR	29
7	The parse service routine scans the command buffer for the expected parameters, builds a parameter descriptor list (PDL) that describes the parameters found, updates the buffer offset, and returns control to the TSO command processor.	Parse	IKJPARSR	18
8	The TSO command processor processes the command according to the parameters received.	Refer to TSO command processors manuals		

Diagram 9. Command Scan and Parse Service Routines (Overview) (Part 2 of 2)



A Input Flags

Flag	Meaning
X'00'	Syntax check
X'80'	No syntax check

B Output Flags

Flag	Meaning	Buffer offset set to:
X'80'	Command name is valid; the remainder of the buffer contains non-separator characters.	First non-separator character following command name.
X'40'	Command name is valid; the remainder of the buffer is empty (contains only separator characters).	End of buffer.
X'20'	Command name is question mark.	Unchanged.
X'10'	Buffer is empty (contains only separator characters).	End of buffer.
X'08'	Command name is syntactically invalid.	Unchanged.
X'04'	Command is the implicit EXEC command.	Determined by the flags above.

Diagram 10. Command Scan Service Routine (Part 1 of 2)

Key	Description	Rontine	Label
	During initialization, issues an unconditional GETMAIN for a command scan work area (CSWORK).	IKJEFP30	IKJSCANR
1	The routine skips separators to the beginning of a command name. If the buffer is empty or if the first character is a question mark, the program exits. Otherwise, the scan continues to the next delimiter.	IKJEFP30 IKJEFP06	SKIPB
	The routine searches the command buffer for DBCS shift-out (X'0E') and shift-in (X'0F') characters and issues informational messages if it finds any DBCS errors in the DBCS string.		FINDSI DBMSG
2	If the high order byte of the flag word is X'00', checks the syntax of the command and, if the first character is '%', sets the 'implicit EXEC flag'. Otherwise the command name must contain valid, enterable characters and end with a delimiter.	IKJEFP40 IKJEFP30	GENSCAN TYPETEST
3	Translates correct command names to uppercase.	IKJEFP40	TRANSX
4	The routine sets the CSOA to indicate the results of the scan and updates the buffer offset as shown in A.	IKJEFP30	CSEXIT80 CSEXIT40 CSEXIT20 CSEXIT10 CSEXIT08
5	Returns to the caller.	IKJEFP30	CSEXIT

Diagram 10. Command Scan Service Routine (Part 2 of 2)

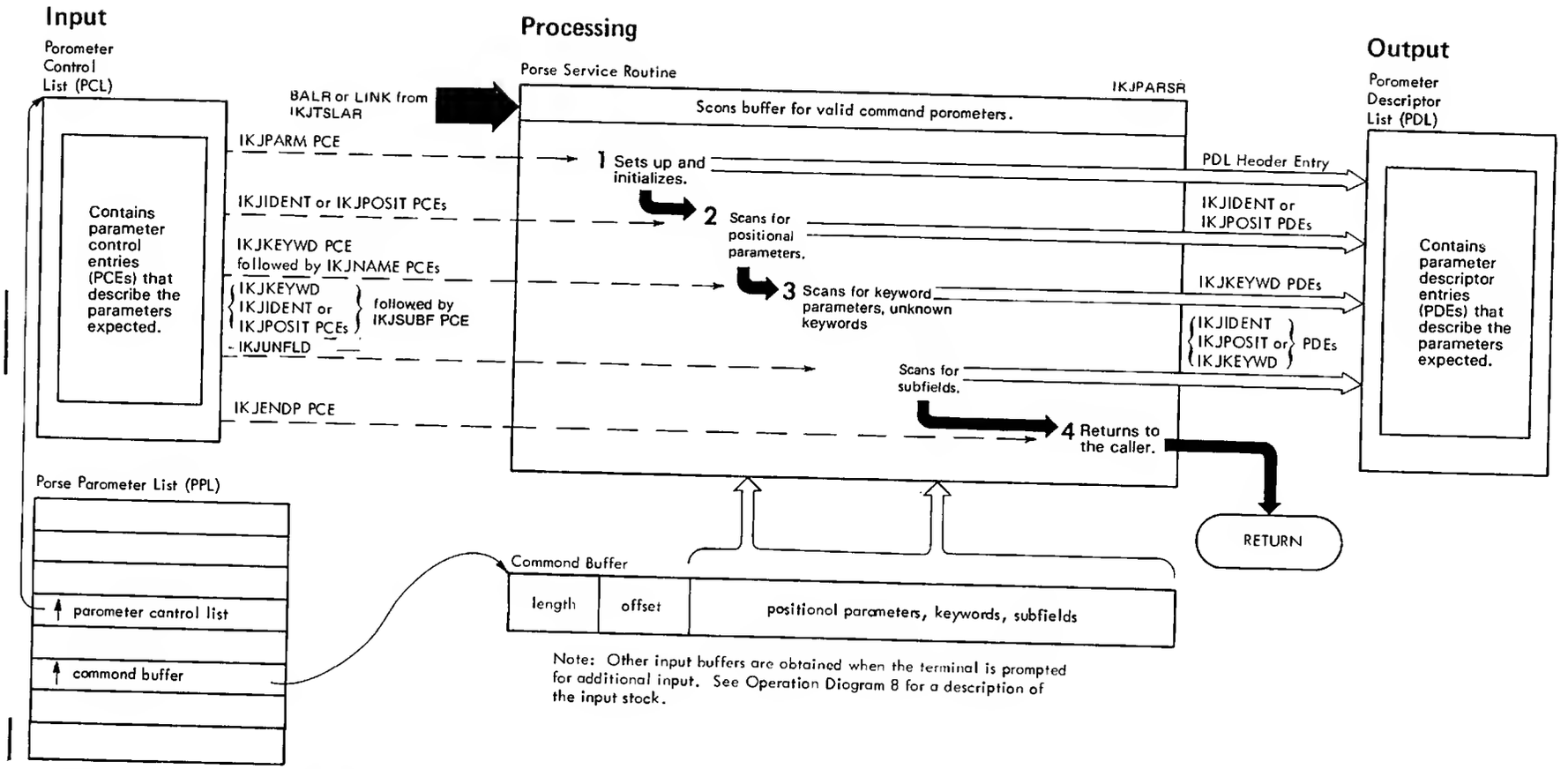


Diagram 11. Parse Service Routine (Part 1 of 2)

Key	Description	Routine	Label	Diagram
	Parse gets PCEs from the PCL and constructs PDEs in the PDL. The operations performed depend upon the type of PCE being processed.			
1	The IKJPARM PCE determines much of what is done during Parse initialization. It names the PCL and PDL (default name is IKJPARMD), and gives the length of the offset in the PCL of the next IKJKEYWD, IKJSUBF, or IKJENDP PCE.	IKJEFP00	IKJPARSR	19
2	The IKJIDENT and IKJPOSIT PCEs determine much of what is done during a scan for positional parameters. The parameters must appear in the command buffer in the same order that their PCEs appear in the PCL. All positional parameters must come before any keyword parameters.	IKJEFP01 IKJEFP00	IDENT POSIT	20
3	The IKJKEYWD, IKJUNFLD, and IKJNAME PCEs determine much of what is done during a scan for keyword parameters. The IKJKEYWD PCE marks the beginning of a keyword field. The IKJNAME PCEs define eligible names for the keyword. IKJUNFLD allows processing of unknown keywords. Keywords may have subfields that include both positional and keyword parameters. The IKJSUBF PCE marks the beginning of a subfield and the end of a previous field.	IKJEFP00	KEYWDP	22
4	The IKJENDP PCE marks the end of the PCL. Parse checks to see if the scan is complete before it returns control to the calling routine.	IKJEFP00	ENDFIELD	22

Diagram 11. Parse Service Routine (Part 2 of 2)

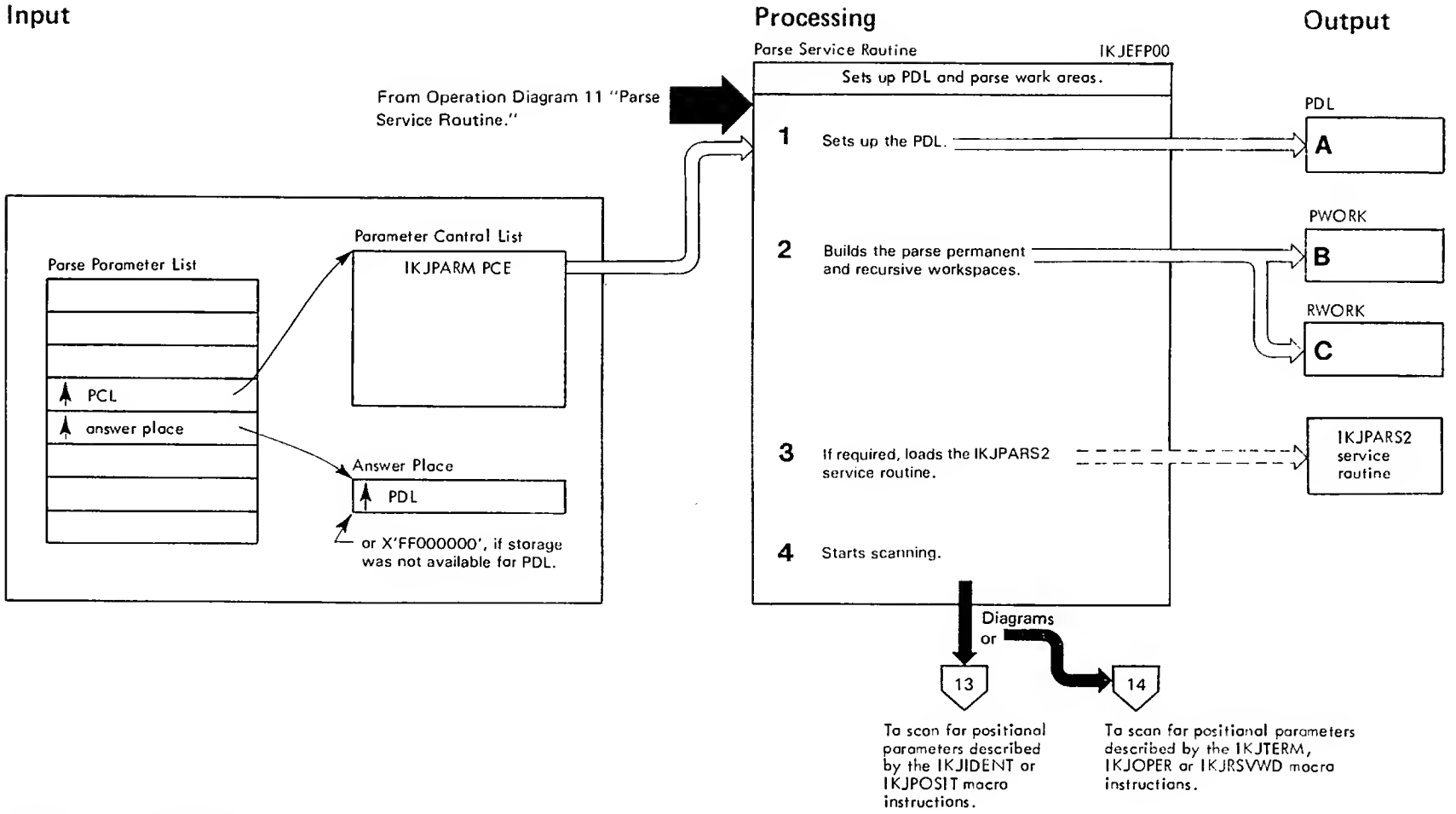


Diagram 12. Parse Initialization (Part 1 of 2)

Key	Description	Routine	Label
1	Parse gets main storage for the PDL from subpool 1. The DSECT for the PDL is named according to the value the IKJPARM PCE specifies. The default name is IKJPARMD.	IKJEFP00 IKJEFP08	IKJPARSR STALOC
2	Obtains storage for the parse work area (PWORK) and first recursive work area (RWORK) from subpool 0. Initializes both work areas with information from the Parse parameter list (PPL).	IKJEFP08	IKJPARSR GETCORE
	Obtains additional recursive work areas each time a subfield is processed.	IKJEFP00	RECURSE
3	If the IKJTERM, IKJOPER, or IKJRSVWD macro instruction is coded, loads the IKJPARS2 service routine.	IKJEFP60	IKJPARS2
4	Prepares to get the next PCE and start the scan.	IKJEFP00	NEXTPCE

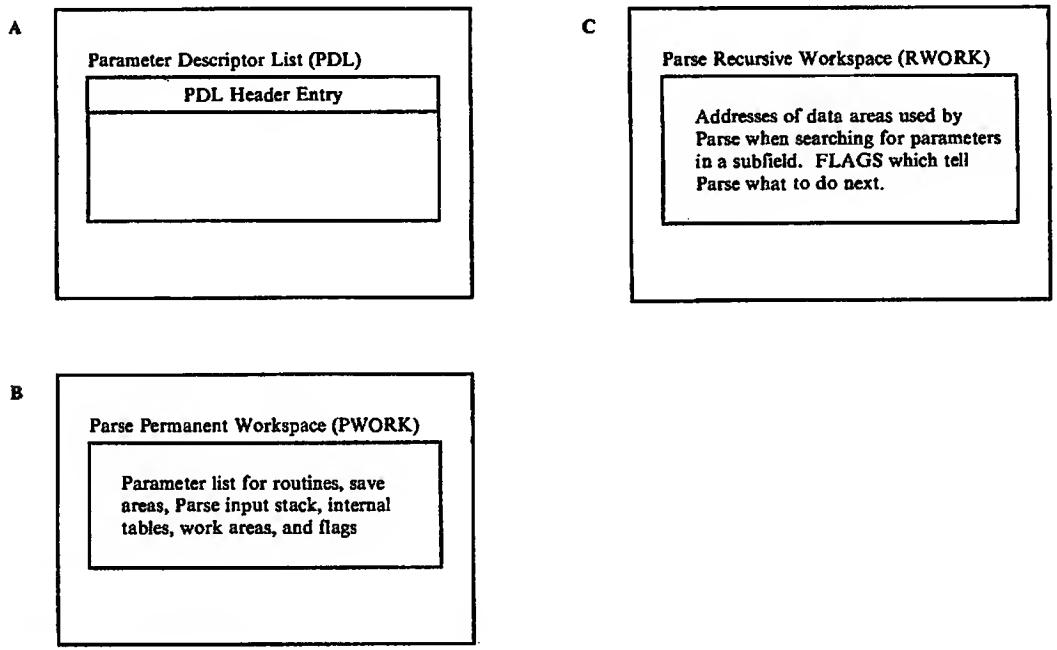


Diagram 12. Parse Initialization (Part 2 of 2)

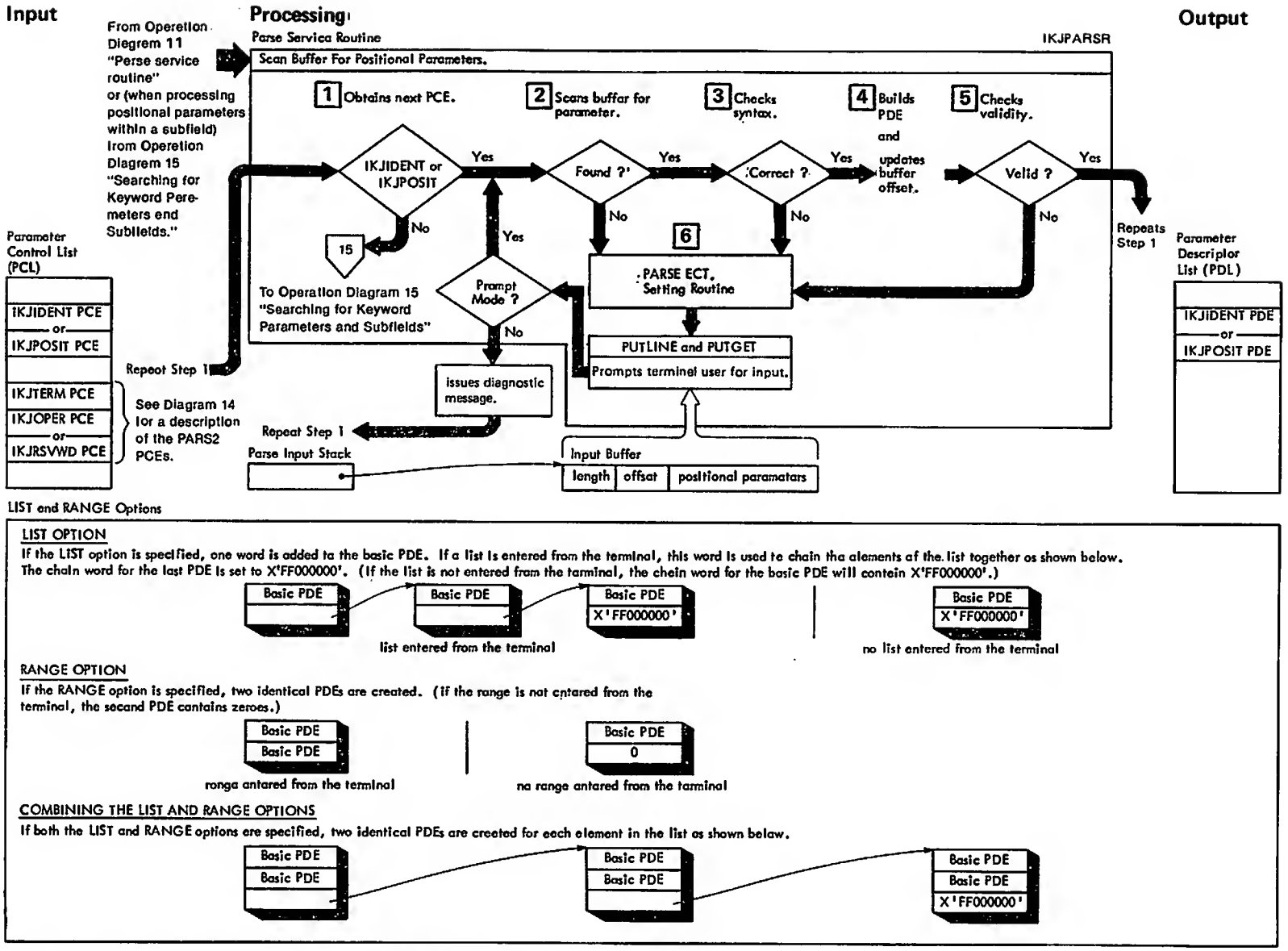


Diagram 13. Searching for IKJPARSR Positional Parameters (Part 1 of 2)

Key Description

1. The IKJIDENT PCE describes a positional parameter in the form of a character string with optional restrictions on the first character, other characters, and length.

   The IKJPOSIT PCE describes a positional parameter which includes a delimiter as part of its syntax. 11 kinds of IKJPOSIT PCEs exist and each has its own processing routine.

2. The buffer is scanned for the parameter.

   If the parameter is missing and the user is in PROMPT mode, checks the PCE to see if it is required or if there is a default.
   - If the parameter is required, prompts the terminal for the parameter.
   - If there is a default, supplies the default.
   - If the parameter is not required, Parse gets the next PCE.
   - If the parameter is required and the user is in NOPROMPT mode, issues a missing message and Parse continues the scan of the input line in invalid mode.

     Note: Invalid mode means that Parse is unable to prompt the user for replacements for missing or invalid data. Parse continues to scan the input line and issues all diagnostic messages. No validity check exits are taken and no PDL is returned to the user.

3. If the parameter is found (or defaulted), checks the parameter for correct syntax.
   - If an error is found and the user is in PROMPT mode, prompts the user to reenter the parameter and repeat step 2.
   - If the user is in NOPROMPT mode and the data is invalid or missing, does not take the validity check exit and issues a diagnostic message.

4. If the parameter is correct, builds a PDE and translates the parameter to upper case. If a list is being processed, repeats step 2 for each element in the list.

5. If the calling routine specified a validity check exit, enters the validity check exit routine. If a range is being processed, passes the complete range to the exit routine. If a list is being processed, passes each element in the list to the exit routine.

   If an error is found, prompts the terminal to reenter the parameter and repeat step 2. If the user is in "no prompt mode," he receives all messages in the list.

6. The ECT setting routine (IKJEFP05) is invoked by IKJEFP00 (mainline PARSE) to set bits in the ECT (Environment Control Table) before PARSE invokes PUTGET. The ECT supplies enough information to PUTGET to allow it to generate a HELP command for the information that PARSE is prompting for when all other informational messages have been issued.

← IKJIDENT PCEs → ← IKJPOSIT PCEs →

		DELIMITER		STRING	
Routine	Label	Routine	Label	Routine	Label
IKJEFP01	IDENT	IKJEFP00	DELIMITER	IKJEFP00	STRING
IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08	SKIP8 PROMPTQ SCANF LISTT RANGE TYPETEST	IKJEFP08 IKJEFP08	SKIPB TYPETEST	IKJEFP08 IKJEFP08 IKJEFP08	PRDMPTQ FINDSI D8CMSG
IKJEFP20	GENSCAN				
IKJEFP20	TRANSQ			IKJEFP20	TRANSQ
IKJEFP09	VCERTN				

← IKJPOSIT PCEs →

VALUE		ADDRESS		PSTRING		USERID		DSNAME/DSTHING		QSTRING		SPACE	
Routine	Label	Routine	Label	Routine	Label	Routine	Label	Routine	Label	Routine	Label	Routine	Label
IKJEFP00	VALUE	IKJEFP00	ADDRESS	IKJEFP00	PSTRING	IKJEFP00	USERID	IKJEFP01	DSNAME	IKJEFP01	QSTRING	IKJEFP01	SPACE
IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08	SKIPB FINDSI D8CMSG LISTT TYPETEST PROMPTQ RANGE	IKJEFP08 IKJEFP06 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08	SKIP8 FINDSI D8CMSG LISTT TYPETEST RANGE	IKJEFP08 IKJEFP06 IKJEFP06 IKJEFP08 IKJEFP08 IKJEFP08	SKIP8 FINDSI D8CMSG PROMPTQ GETCORE WRITER1	IKJEFP08 IKJEFP08 IKJEFP06 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP09	SKIP8 FINDSI D8CMSG LISTT TYPETEST SCANF GETCORE PROMPTQ WRITE2G	IKJEFP08 IKJEFP06 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP09	SKIP8 FINDSI D8CMSG LISTT SCANF TYPETEST PRDMPTQ GETCORE WRITER2G	IKJEFP08 IKJEFP06 IKJEFP08 IKJEFP08 IKJEFP08 IKJEFP08	SKIP8 FINDSI D8CMSG PROMPTQ GETCORE WRITER1		
						IKJEFP20	GENSCAN	IKJEFP20	GENSCAN				
IKJEFP20	TRANSQ	IKJEFP02	TRANSQ	IKJEFP20	TRANSQ	IKJEFP20	TRANSQ	IKJEFP20	TRANSQ	IKJEFP20	TRANSQ		
IKJEFP09	VCERTN	IKJEFP09	VCERTN	IKJEFP09	VCERTN	IKJEFP09	VCERTN	IKJEFP09	VCERTN	IKJEFP09	VCERTN		

Diagram 13. Searching for IKJPARSR Positional Parameters (Part 2 of 2)

LIST and RANGE Options

LIST OPTION

If the LIST option is specified, adds one word to the basic PDE. If a list is entered from the terminal, uses this word to chain the elements of the list together as shown below. Sets the chain words for the last PDE to X'FF000000'. (If the list is not entered from the terminal, the chain word for the basic PDE contains X'FF000000'.)

RANGE OPTION

If the RANGE option is specified, creates two identical PDEs. (If the range is not entered from the terminal, the second PDE contains zeroes.)

COMBINING THE LIST AND RANGE OPTIONS

If both the LIST and RANGE options are specified, creates two identical PDEs for each element in the list as shown below.

Note 1: The IKJRSVWD macro instruction can also be chained to the IKJTERM macro and the IKJOPER macro to define the beginning of a list of reserved words.

Diagram 14. Searching for IKJPARS2 Positional Parameters (Part 1 of 2)

Key Description

1 • The IKJTERM PCE describes a positional parameter that can be entered as a constant, variable or statement number.
• The IKJOPER PCE describes a positional parameter that can be entered as an expression.
• The IKJRSVWD PCE describes a positional parameter that can be entered as a reserved word (also see note 1).

2 The buffer is scanned for the parameter.

If the parameter is missing, checks the PCE to see if it is required or if there is a default.

- If the parameter is required and the user is in PROMPT mode, prompts the terminal for the parameter.
- If there is a default, supplies the default.
- If the parameter is not required, gets the next PCE.
- If the parameter is required and the user is in NOPROMPT mode, issues a message and continues the scan of the input line in invalid mode.

Note: Invalid mode means that Parse is unable to prompt the user for replacements for missing or invalid data. Parse continues to scan the input line and issues all diagnostic messages. No validity check exits are taken and no PDL is returned to the user.

3 If the parameter is found (or defaulted), checks the parameter for the correct syntax.
• If an error is found and the user is in PROMPT mode, prompts the user to reenter the parameter and repeats Step 2.
• If the user is in NOPROMPT mode and the data is invalid or missing, does not take the validity check exit and issues a diagnostic message.

4 If the parameter is correct, builds a PDE and translates the parameter to upper case. If a list is being processed, repeats Step 2 for each element in the list.

5 If the calling routine specified a validity check exit, enters the validity check exit routine. If a range is being processed, passes the complete range to the exit routine. If a list is being processed, passes each element in the list to the exit routine.

6 IKJEFP00 (mainline PARSE) invokes the ECT setting routine (IKJEFP05) to set bits in the ECT (Environment Control Table) before PARSE invokes PUTGET. The ECT supplies enough information to PUTGET to allow it to generate a HELP command for the information that PARSE is prompting for when all other informational messages have been issued.

IKJPARSR Routines Used by IKJPARS2

CLEANUP	- Free care, exit
GENSCAN	- Parameter scan routine
GETCORE	- Obtain storage (release before exit)
LISTT	- Test for LIST entered
NAMESKP3	- Skip to next PCE routine
NEXTPCE	- Go to next PCE routine
PARS2ENT	- Entry into IKJPARSR from IKJPARS2 when subroutine functions are required
POSITX	- Add PDE to PDL routine
PROMPTQ	- Prompt with 'ENTER...' routine
PSTMIMSG	- Ending parenthesis assumed message
PUSHI	- Push the stack routine
RANGE	- Test for a RANGE entered
SCANF	- Pop the stack routine
SKIPB	- Skip blanks routine
STALOC	- Allocate storage in subpool 1
SYSR1	- Set up 'INVALID' message, then prompt
TRANSQ	- Translate to uppercase routine
TYPETEST	- Test for character type routine

Other Routines Used by IKJPARS2

FINDSI	- Search for shift-in character

Diagram 14. Searching for IKJPARS2 Positional Parameters (Part 2 of 2)

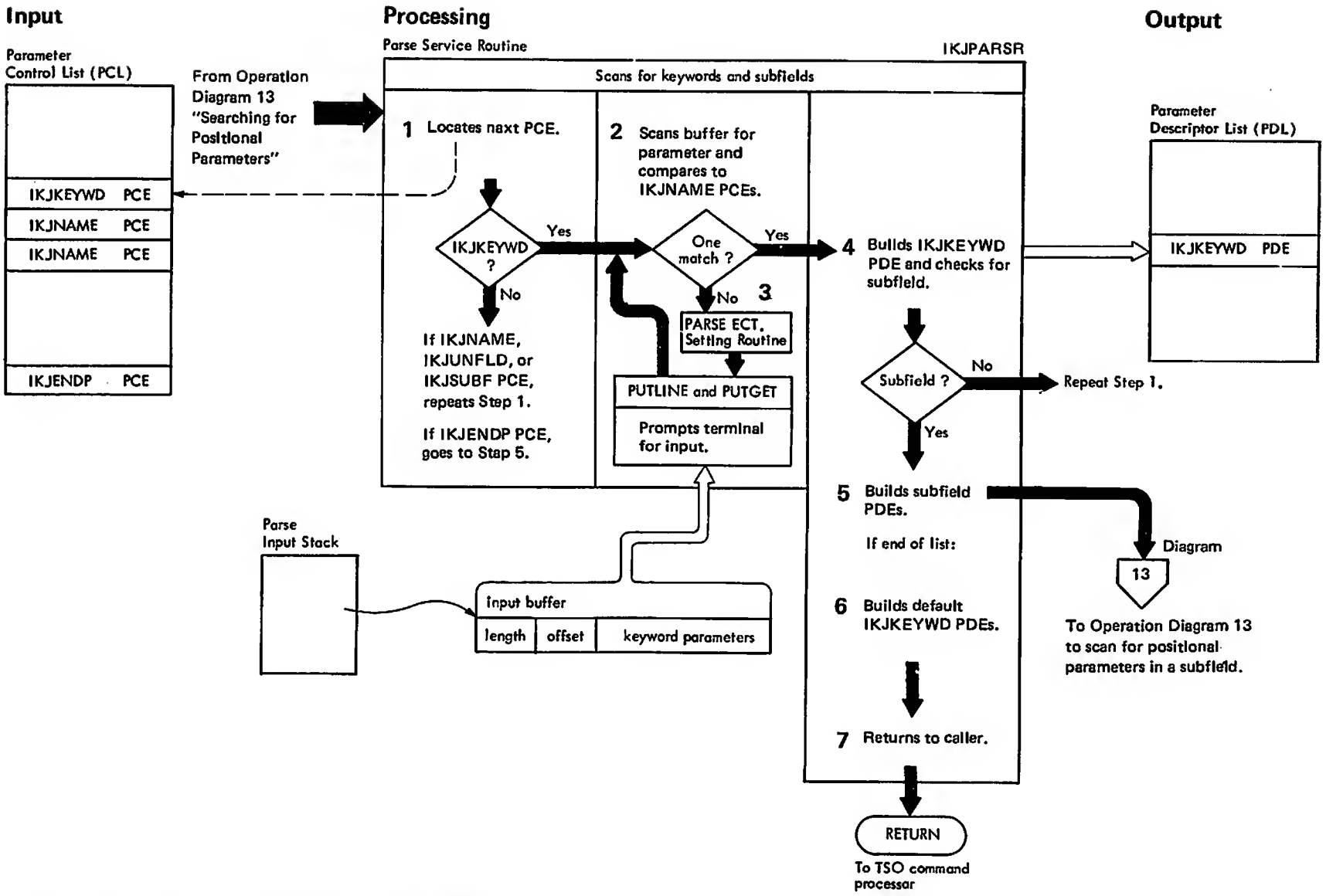


Diagram 15. Searching for Keyword Parameters and Subfields (Part 1 of 2)

Key	Description	Routine	Label
1	The IKJKEYWD PCE marks the beginning of a keyword field and specifies optioos for that field.	IKJEFP00	KEYWDP
2	The routine scans the buffer for a keyword and compares the result to the names specified by the IKJNAME PCEs.	IKJEFP01	KEYWD
3	IKJEFP00 (mainline PARSE) invokes the ECT setting routine (IKJEFP05) to set bits in the ECT (Eovirooment Control Table) before PARSE invokes PUTGET. The ECT supplies enough informatioo to PUTGET to allow it to generate a HELP command for the information that PARSE is promptiog for wheo all other ioformatiooal message have been issued.	IKJEFP01 IKJEFP05	KEYWDNAM
4	If a match is found, the routine builds IKJKEYWD PDE aod checks the IKJNAME PCE for a subfield.	IKJEFP01	KEYWDNAM
	• If the routine does oot find a match, it prompts the termioal to reeoter the parameter.	IKJEFP08	PROMPTQ
	• If it finds more than one match, it erases all previous PDEs built for the keyword, and prompts tfie termioal to reeoter the parameter (see "Erasiog PDEs" in Section 1).	IKJEFP00 IKJEFP08	KEYWDER4 PROMPTQ
	• If no keyword match is found, Parse checks for an IKJUNFLD field. If found, the verify exit is invoked and the field is processed.		
5	If the keyword name has a subfield, Parse interrupts the scan for the keyword field and processes the subfield in the same way that it would process a field. When another IKJSUBF or an IKJENDP PCE is reached, Parse resumes the scan for the keyword field. If the subfield is iovalid or missing, Parse prompts with the user-entered keyword followed by a left pareothesis. Only the subfield need be entered in respoose. A right parenthesis should oot be eotered.	IKJEFP01 IKJEFP00 IKJEFP00	KEYWDSUB ENDFIELD KEYWDP
6	When all of the keyword fields have been scanned, Parse checks each IKJKEYWD PCE to see if the correspooding PDE has been built.	IKJEFP00	KEYWDP
	If not, it supplies a default value, if one is specified in the IKJKEYWD PCE. (Keyword parameters are never required.)	IKJEFP08	PROMPTQ
7	When the routine reaches an IKJENDP PCE, Parse checks to see if it completed the scan.	IKJEFP00	ENDFIELD
	• If the scan reached the end of the buffer, Parse returns control to the calling routioe.	IKJEFP00	ENDEX1
	• If the scan did not reach the end of the buffer, Parse writes an "extraneous information" message, or, if there are keyword PCE's io the PCL, it writes an "iovalid keyword" message, and returns control to the calling routine.	IKJEFP00	ENDEX2
	• If a verify exit was called, Parse calls the exit a final time for cleanup purposes.		

Diagram 15. Searching for Keyword Parameters and Subfields (Part 2 of 2)

Program Organization

This section describes the organization of command scan and parse. It gives information about the hierarchy of the load modules, the assembly modules, and the control sections that constitute each program.

The module operation information outlines the processing operations that occur within each parse and command scan module.

Program Hierarchy

IKJPARSR -- Parse Load Module

IKJEFP00 -- Parse mainline routine. This module has two control sections: IKJEFP00 and IKJEFP01.

IKJEFP03 -- Parse address PCE processor

IKJEFP05 -- The ECT setting routine

IKJEFP06 -- Subroutines common to command scan and parse for DBCS processing

IKJEFP08 -- Parse mainline subroutines

IKJEFP10 -- Parse messages

IKJEFP20 -- Parse syntax checking routine

IKJPARS2 is an assembly module that IKJPARSR loads when an IKJTERM, IKJOPER or IKJRSVWD macro instruction is executed. IKJPARS2 has three control sections: IKJEFP06, IKJEFP10, and IKJEFP60.

IKJSCANR -- Command Scan Load Module

IKJEFP30 -- Command scan
IKJEFP40 -- Command scan syntax checking routine
IKJEFP06 -- Subroutines common to command scan and parse
IKJEFP10 -- Parse messages

Module Operation

The following descriptions outline the processing operations in each executable module of the parse and command scan service routines.

IKJEFP00 -- Parse Mainline Routine

This module has two control sections:

IKJEFP00
IKJEFP01

IKJEFP00 scans the buffer for command parameters, checks their syntax, optionally translates them to uppercase, optionally checks their validity, builds a PDL consisting of PDEs that describe the parameter found, and returns to the calling program.

IKJEFP03 -- Parse Address PCE Processor

IKJEFP03 checks the syntax and determines the type of each address entered.

IKJEFP05 -- ECT Setting Routine

IKJEFP05 sets the bits in the ECT that contain information needed by PUTGET to generate a HELP command. The HELP command responds to PARSE prompting after all other informational messages have been issued.

IKJEFP06 -- Subroutines Common to Command Scan and Parse

Contains the following subroutines:

FINDSI — Searches the command buffer for a shift-in character (X'0F') to match the shift-out character (X'0E') found by command scan or parse.

If a matching shift-out and shift-in pair is found, FINDSI sets a return code of zero. If any of the following error conditions is found, a flag bit is set in ERROR-FLAGS to indicate the nature of the error and a non-zero return code is set:

- Missing shift-in character
- OUT-OF-RANGE DBCS character
- Odd number of bytes in the DBCS
- Nested shift-out character in the DBCS.

DBCMSG — Sets up and issues messages for each error identified in the MSG-FLAGS word. DBCMSG returns a non-zero return code reflecting the DBCS error condition.

IKJEFP08 -- Parse Mainline Subroutines

IKJEFP08 consists of the following routines:

STALOC -- Allocates main storage for PDL and prompt data.

GETCORE -- Gets more storage.

SKIPB -- Updates the buffer offset to the first non-separator character.

PROMPTQ -- If prompt was specified, prompts the terminal user and saves the address of the command buffer and message ID.

SCANF -- Checks for end of stack, removes the last element put on the push-down stack, and issues a FREEMAIN instruction for the previous stack.

LISTT -- Checks for a list. If a list was entered, gets the address of the first element.

RANGE -- Checks to see if a range is possible.

TYPETEST -- Tests the current character against a selected mask.

VCERTN -- Sets up the linkage to the user-supplied validity check exit routine.

WRITER1 -- Writes informational messages.

WRITER2 -- If necessary, prompts the terminal user for input or checks for default. Takes any new data and places it in storage allocated by STALOC.

UNKSERCH -- Checks to see if an unknown PCE exists for the type of parameter being checked.

FINDUPCE -- Searches the PCL for the first occurrence of an unknown PCE.

CALLVERX -- Calls the unknown parameter verify routine.

VERRCHK -- Checks the verify exit return code.

PUTINVLD -- Issues an invalid parameter message.

CHKSECLV -- Checks for a second level message.

SKIPLIST -- Skips to the end of the list when requested.

BILDPCES -- Modifies the unknown PCE to represent a keyword and name PCE for the verified operand.

IKJEFP20 -- Parse Scan Module

IKJEFP20 consists of the following routines:

GENSCAN -- Is a generalized scan routine that checks the syntax of the command buffer parameter according to control information set up by the invoker or parse.

TRANSQ -- Is a translate routine that translates lowercase alphabetic characters to uppercase.

TRANSX -- If the parameter is known not to be defaulted, translates lowercase alphabetic characters to uppercase.

Note that if characters are in a DBCS string, neither TRANSQ nor TRANSX translates the hexadecimal values of any lowercase characters to their uppercase counterparts. For example, if X'81' (a lowercase "a") does not appear within in a DBCS string, it is translated to X'C1' (an uppercase "A"). If X'81' appears within a DBCS string, it is not translated.

IKJEFP30 -- Command Scan Mainline Routine

IKJEFP30 searches the buffer for a valid command name, optionally checks the syntax of the command name, translates the command name to uppercase, indicates whether parameters follow the command name, updates buffer offset, and returns to the calling program.

IKJEFP40 -- Command Scan Syntax Checking Routine

IKJEFP40 checks syntax for IKJEFP30.

IKJEFP60 -- IKJPARS2 Service Routine

1. Entry point IKJEFP40 checks the syntax of reserved word parameters specified on the IKJRSVWD macro instruction.
2. Entry point IKJEFP50 checks the syntax of expression parameters specified on the IKJOPER macro instruction.
3. Entry point IKJEFP60 checks the syntax of constant, variable, or statement number parameters specified on the IKJTERM macro instruction.

Directory

This table will help you find the appropriate module operation description or assembly listing.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
ADDRESS	Positional Address	IKJPARSR	IKJEFP03	IKJEFP03	Scans buffer for address - Prompts for missing data. Identifies address type. Builds temporary PDE. Checks for possible list. Branches to ILLADR if illegal address. Otherwise returns to NEXTPCE.
BILDPCES	Build PCE	IKJPARSR	IKJEFP08	IKJEFP08	Modifies the unknown PCE to represent a keyword and name PCE for the verify operand.
BUMP	Bump the Input Stack	IKJPARS2	----	IKJEFP60	Tests for more data. Takes off the last element put on the push-down stack, and continues processing, if more data. Returns +0 on no more data, +4 on more data.
CALLVERX	Call Validity Check Exit	IKJPARSR	IKJEFP08	IKJEFP08	Calls the unknown parameter verify routine.
CHKSECLV	Check Message	IKJPARSR	IKJEFP08	IKJEFP08	Checks for a second level message.
CLEANUP	Cleanup	IKJPARSR	IKJEFP08	IKJEFP09	Frees all storage obtained by Parse when terminal error occurs. Places X'FF000000' in answer place.
DBCMSG	Double Byte Message Writing Routine	IKJSCANR IKJPARSR IKJPARS2	IKJEFP06	IKJEFP06	Issues one or more informational messages describing DBCS errors that occurred.
DELIMITR	Positional	IKJPARSR	IKJEFP00	JKJEFP00	Scans buffer for next self-defining delimiter and sets switch if invalid.
DSNAME	Positional Data Set Name	IKJPARSR	IKJEFP00	IKJEFP01	Scans buffer for DSNAME, DSTHING, or DDNAME. Builds temporary PDE. Scans for JOBNAME, volume serial number, and DSNAME passwords. Also prefixes USERID to DSNAME, if required.
ENDFIELD	End-of-Field Processing Routine	IKJPARSR	IKJEFP00	IKJEFP00	Entered at end of field -- Checks for extraneous data in buffer or subfield and writes error message. Entered when erasing PDEs -- Releases current RWORK and gets next RWORK. Entered at end of subfields -- all functions executed for other entries.
EXIT	Final Exit	IKJPARSR	IKJEFP00	IKJEFP00	Frees PWORK AND RWORK. Updates buffer offset to point one character past the last character scanned.
FINDSI	Find Shift-in Character	IKJSCANR IKJPARSR IKJPARS2	IKJEFP06	IKJEFP06	Invoked after a shift-out character was found. Searches the command buffer for a shift-in character and checks the DBCS string for invalid characters, odd length, and missing shift-in character.
FINDUPCE	Search PCL for PCE	IKJPARSR	IKJEFP08	IKJEFP08	Searches the PCE for the first occurrence of an unknown PCE.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
FREECORE	Issue FREEMAIN	IKJPARS2	----	IKJEFP40	Issues FREEMAIN to release storage.
GENSCAN	General Scan	IKJPARSR IKJSCANR	IKJEFP20 IKJEFP40	IKJEFP20 IKJEFP40	Scans buffer for character string.
GETCORE	Issue Getmain	IKJPARSR	IKJEFP08	IKJEFP08	Gets more storage.
IDENT		IKJPARSR	IKJEFP00	IKJEFP01	Scans for buffer positional parameter.
IKJEFP05	ECT Setting Routine	IKJPARSR	IKJEFP05	IKJEFP05	Supplies the information to PUTGET that enables it to generate a HELP command in response to a '?'.
IKJEFP40	IKJRSVWD Processing	IKJPARS2	IKJEFP60	IKJEFP40	Processes the reserved word parameters.
IKJEFP50	IKJOPER Processing	IKJPARS2	IKJEFP60	IKJEFP50	Processes the expression parameter.
IKJEFP60	IKJTERM Processing	IKJPARS2	IKJEFP60	IKJEFP60	Processes a constant, variable, or statement number parameter.
IKJPARSR	Parse Service Routine	IKJPARSR	IKJEFP00	IKJEFP00	Gets storage for PWORK and RWORK.
IKJPARS2	Parse 2 Service Routine	IKJPARS2	IKJEFP60	IKJEFP50	Processes an IKJTERM, IKJOPER, or IKJRSVWD macro instruction when loaded by IKJPARS.
IKJSCANR	Command Scan	IKJSCANR	IKJEFP30	IKJEFP30	Scans buffer for command name.
KEYWD	Keyword Scan	IKJPARSR	IKJEFP00	IKJEFP01	Scans buffer for keywords and subfields.
KEYWDP	Keyword Processor	IKJPARSR	IKJEFP00	IKJEFP00	Entered before keyword field is scanned. Indicates keyword scan and gives control to scan routine. Entered after keyword field is scanned--checks to see if PDE was built. If not, gets default and gives control to keyword scan. If so, gives control to main control. Entered when erasing PDEs--zeros the keyword PDE, calculates the next PCE address.
LISTT	List	IKJPARSR	IKJEFP08	IKJEFP08	Checks for list, if one was entered, gets address of first element.
MSNGMSG	Missing Message	IKJPARSR	IKJEFP08	IKJEFP09	Issues "MISSING" message when an "ENTER" or "ENTER PASSWORD" prompt message was attempted during no-prompt mode.
NAMESKP	Name Skip	IKJPARSR	IKJEFP00	IKJEFP00	Entered after keyword field scan--Skips IKJNAME PCEs. When erasing IKJKEYWD PDEs--Checks IKJNAME PCE for subfield. If found, returns to RECURSE. If not, returns to NEXTPCE.
NEXTPCE	PCI Locating Routine	IKJPARSR	IKJEFP00	IKJEFP00	Locates the next PCE and branches to the appropriate PCE processing routine.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
POSIT	Positional Processor	IKJPARSR	IKJEFP00	IKJEFP00	Gives control to a 2nd level positional parameter processor.
POSITERS	Positional PDE Erase	IKJPARSR	IKJEFP00	IKJEFP01	Erases a PDE for a positional parameter.
POSITX	Positional Exit	IKJPARSR	IKJEFP00	IKJEFP01	Checks for and processes a range or list. Translates to uppercase if string, pstring, gstring, value, or ident macro was used.
PROMPTQ	Prompt Default	IKJPARSR	IKJEFP08	IKJEFP08	If prompt was specified--Prompts terminal. Saves address of command buffer and message ID.
PSTRIMSG	Parenthesis Assumed Message	IKJPARSR	IKJEFP00	IKJEFP00	If a right parenthesis is not found, writes a message indicating it is assumed.
PSTRING	Positional Parenthesized String	IKJPARSR	IKJEFP00	IKJEFP00	Scans buffer for string in parentheses and checks for embedded parentheses.
PUSHI	Push Input Stack	IKJPARSR	IKJEFP08	IKJEFP08	Saves current buffer pointer and end-of-buffer pointer on input stack. Gets more storage when stack is full.
PUTINVLD	Issue Message	IKJPARSR	IKJEFP08	IKJEFP08	Issues a message that a parameter is invalid.
QSTRING	Positional Quoted String	IKJPARSR	IKJEFP00	IKJEFP01	Scans buffer to next quote.
RANGE	Range	IKJPARSR	IKJEFP00	IKJEFP09	Checks to see if a range is possible.
RECURSE	Subfield Processing	IKJPARSR	IKJEFP00	IKJEFP00	Gets main storage for RWORK.
SCANF	Pop the Stack	IKJPARSR	IKJEFP08	IKJEFP08	Checks for end of stack. Removes last element put on the push-down stack. Issues freemain for previous stack.
SCANPOP	Pop Input Stack	IKJPARSR	IKJEFP08	IKJEFP08	Gets last element from input stack.
SKIPB	Skip Separators	IKJPARSR	IKJEFP08	IKJEFP08	Updates buffer offset to first non-separator character.
SKIPLIST	Skip List	IKJPARSR	IKJEFP08	IKJEFP08	Skips to the end of the list when requested.
STALOC	Storage Allocate	IKJPARS	IKJEFP08	IKJEFP08	Allocates main storage for PDL and prompt data.
STRING	Positional String	IKJPARSR	IKJEFP00	IKJEFP00	Prompts for data if switch is set by delimiter routine. Scans buffer for character string.
SYSR1	Error Handling	IKJPARSR	IKJEFP08	IKJEFP09	Calculates length of invalid data. Builds informational message segments.
TERMOCK	Check TERM PCE	IKJPARS2	---	IKJEFP40	Checks and test the IKJTERM PCE.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
TRANSQ	Translate	IKJPARSR and IKJSCANR	IKJEFP20 IKJEFP40	IKJEFP20 IKJEFP40	Translates lowercase alphabetic characters to uppercase.
TYPETEST	Character Type Test	IKJPARSR	IKJEFP00	IKJEFP09	Tests current character against selected mask.
UNKSERCH	Unknown PCE Search	IKJPARSR	IKJEFP08	IKJEFP08	Checks to see if an unknown PCE exists for the type of parameter being processed.
USERID	Positional Userid	IKJPARSR	IKJEFP00	IKJEFP00	Scans buffer for user ID and password.
VALUE	Positional Value	IKJPARSR	IKJEFP00	IKJEFP00	Checks type character.
VCERTN	Validity Check Exit	IKJPARSR	IKJEFP08	IKJEFP09	Sets up linkage to user-supplied validity check exit routine.
VERRCHK	Verify Return Code	IKJPARSR	IKJEFP08	IKJEFP08	Checks the verify exit return code.
WRITER1	Informational Messages	IKJPARSR	IKJEFP08	IKJEFP08	Writes informational messages.
WRITER2	Prompt Messages	IKJPARSR	IKJEFP08	IKJEFP09	Prompts terminal for input if necessary, or checks for default. Takes any new data and places it in storage allocated by STALOC.

Diagnostic Aids

This section contains a summary of return codes and their meanings.

Routine	Return Code Hexadecimal	Meaning
IKJEFP00	00	Successful.
	04	Unable to prompt, parameter missing.
	08	Processing interrupted by attention.
	0C	Invalid parameters passed to parse by command processor.
	10	No space available for PDL.
	14	Validity check routine requested termination.
	18	Invalid parameters passed to the IKJTERM, IKJOPER, or IKJRSVWD macros.
	1C	Line drop was detected by the I/O routines.
	20	Verify exit routine requested termination.
	24	Invalid DBCS character found.
	28	Odd number of bytes in DBCS.
	2C	Shift-out found with no corresponding shift-in.
	30	Nested shift-out character found.
IKJEFP40	--	Return codes are not used. The address returned to in the calling program is determined by the result of IKJEFP40 processing.
IKJEFP30	00	Successful.
	04	The CSPL contains invalid parameters. (The output area and command buffer offset are unchanged.)

Chapter 5. Dynamic Allocation Interface Routine

The dynamic allocation interface routine (DAIR) allocates and frees the data sets that the terminal monitor program, the TSO/E command processors, and other TSO/E problem programs need. In general, DAIR obtains information about a data set and, if necessary, invokes dynamic allocation routines to perform the requested operation.

DAIR invokes dynamic allocation to do the following operations.

- Find the current status of a data set.
- Allocate a data set.
- Build and store lists of data set attributes.
- Assign attributes to data sets.
- Free a data set or attribute list.
- Concatenate data sets.
- Deconcatenate data sets.

The DAIR service routine resides below 16 megabytes in virtual storage and executes in either 24-bit or 31-bit addressing mode. If in 31-bit addressing mode, DAIR can process input above 16 megabytes.

Method of Operation

This section describes DAIR's method of operation. DAIR provides an interface to dynamic allocation, which changes the allocation of data sets needed by the terminal monitor program, TSO/E command processor, and other TSO/E problem programs. Note, however, that the dynamic allocation routines provide the actual changes to the control blocks that make these changes.

Entry to DAIR

A LINK macro instruction to entry point IEFDB4D0 can invoke DAIR. The CALLTSSR macro can also attach, load, and call DAIR. In addition, a user program can link-edit to DAIR.

At entry, register 1 points to the DAIR parameter list (DAPL). The DAIR parameter list contains:

- The address of the user profile table (UPT).
- The address of the environment control table (ECT).
- The address of the calling program's event control block (ECB).
- The address of the protected step control block (PSCB).
- The address of the DAIR parameter block. DAIR parameter block names are in the form DAPBnn, where nn is the last byte of the DAIR entry code.

The DAIR parameter block is the major input to DAIR. The first two bytes contain an entry code (for example X'0008') that defines the operation requested. The remaining bytes contain information, such as the ddname for the data set, the dsname for the data set, and whether the user ID must be prefixed to the dsname.

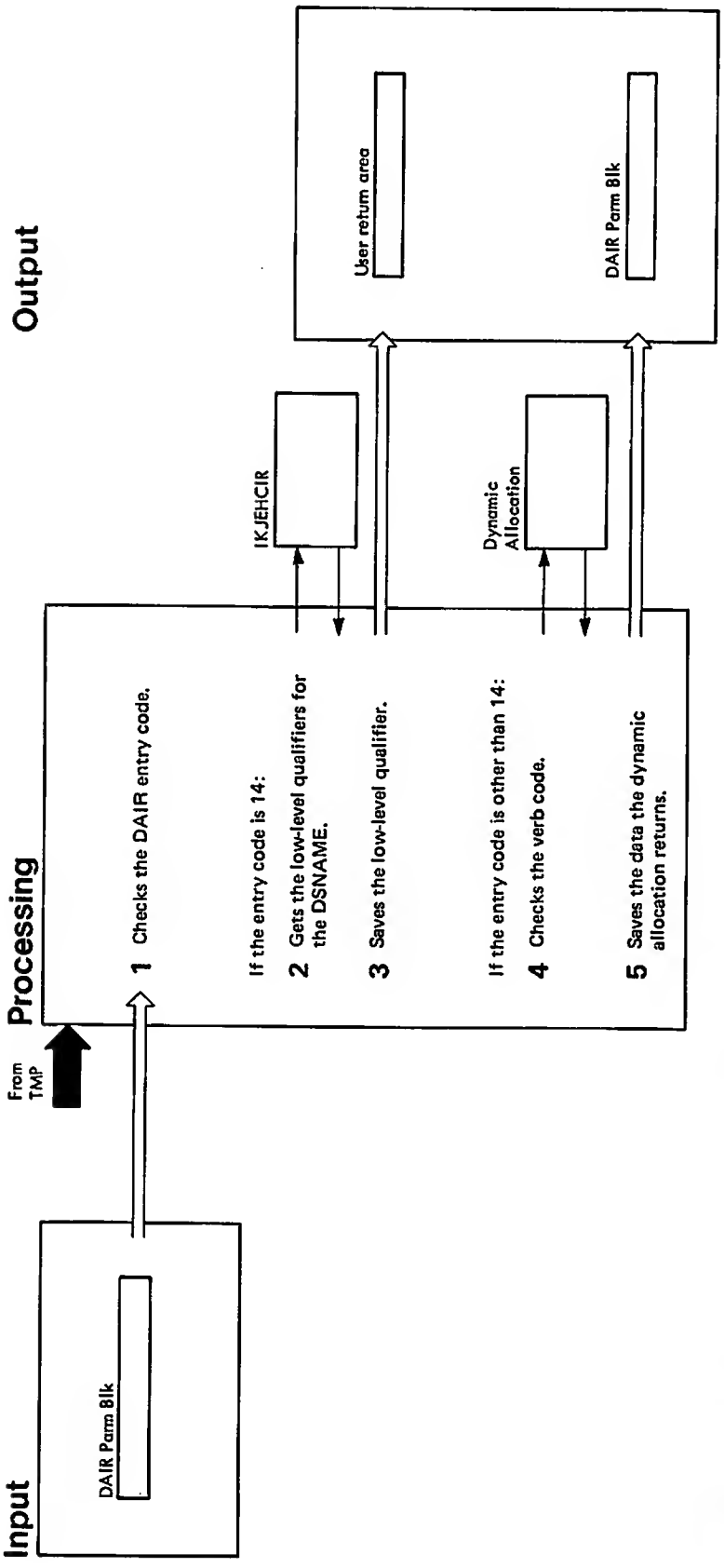


Diagram 16. Dynamic Allocation Interface Routine (Part 1 of 2)

Key	Description
1	Checks the DAIR entry code. If the code is 14, goes to Step 2. Otherwise, goes to Step 4.
	Entry code 14:
2	Builds a catalog information routine (CIR) parameter list and passes control to IKJEHCIR. IKJEHCIR returns low-level qualifiers for the DSNAME to DAIR.
3	Saves the low-level qualifiers in the user return area and returns control to the TMP.
	Entry code other than 14:
4	Uses dynamic allocation to check the verb code and performs the indicated operation.
5	Saves the returned data in the DAIR parameter block. Returns control to the TMP.

Diagram 16. Dynamic Allocation Interface Routine (Part 2 of 2)

Program Organization

This section describes the organization of the dynamic allocation interface routine (DAIR). It contains information about the hierarchy of DAIR and the processing that occurs within DAIR. For a summary of the functions of each DAIR subroutine, see the Directory.

Program Hierarchy

The DAIR service routine has only one load module, IEFDB4D0. The load module includes the following major routines:

DAIR00 -	Searches the data set association block (DSAB) chain for information about a data set.
DAIR04 -	Searches the DSAB chain and system catalog, if necessary, for information about a data set.
DAIR08 -	Allocates a data set by dsname.
DAIR0C -	Concatenates data sets by ddname.
DAIR10 -	Deconcatenates data sets by ddname.
DAIR14 -	Searches the system catalog for qualifiers for a dsname.
DAIR18 -	Frees a data set.
DAIR1C -	Allocates the user's terminal as an I/O device.
DAIR24 -	Allocates a data set by ddname or dsname.
DAIR28 -	Performs a list of operations.
DAIR2C -	Marks DSAB entries not in use for the specified task.
DAIR30 -	Allocates a SYSOUT data set.
DAIR34 -	Allocates dummy data definition statements with DCB parameters.
DSNCHECK -	Checks the validity of the DSNAME and, if necessary, prefixes the user ID to the DSNAME.

Module Operation

The following descriptions outline the processing operations in DAIR.

IEFDB4D0 -- DAIR

Dynamic allocation routines perform the following functions:

- Allocate a data set.
- Free a data set.
- Concatenate a group of data sets.
- Deconcatenate a group of data sets.
- Allocate or free a dummy data definition statement with associated DCB parameters.

Directory

This directory contains information to help you find the appropriate program description or assembler listing. The directory correlates information from three sources:

- The source code
- The executable load modules
- This manual.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
IEFBD4D0	Control Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Routes control to the DAIR routine that corresponds to the entry code.
DAIR00	Code X'00' Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Searches for a data set in the DSAB chain.
DAIR04	Code X'04' Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Searches for a data set in the DSAB chain and system catalog.
DAIR08	Code X'08' Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Allocates a data set by dsname.
DAIR0C	Code X'0C' Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Concatenates data sets by ddname.
DAIR10	Code X'10'	IEFDB4D0	IEFDB4D0	IEFDB4D0	Deconcatenates data sets by ddname.
DAIR14	Code X'14' Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Searches for dsname qualifiers.
DAIR18	Code X'18' Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Frees a data set.
DAIR1C	Code X'1C' Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Allocates the user's terminal as an I/O device.
DAIR24	Code X'24' Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Allocates a data set by ddname or dsname.
DAIR30	Code X'30'	IEFDB4D0	IEFDB4D0	IEFDB4D0	Allocates a SYSOUT data set.
DAIR34	Code X'34' Routine	IEFDB4D0	IEFDB4D0	IEFDB4D0	Allocates DUMMY DD statements.
DSNCHECK	------	IEFDB4D0	IEFDB4D0	IEFDB4D0	Checks the DSNAME for validity and, if necessary, prefixes the USER ID.

Diagnostic Aids

This section contains a summary of DAIR return codes and their meanings.

Return Code Hexadecimal	Meaning
0	DAIR completed successfully.
4	The parameter list passed to DAIR was invalid.
8	An error occurred in a catalog management routine; the catalog management error code is stored in the CTRC field of the DAIR parameter block.
0C	An error occurred in dynamic allocation; the dynamic allocation error code is stored in the DARC field of the DAIR parameter block.
10	Allocation could not free the resources held in anticipation of their reuse to decrease their number to the control value; - OR - Concurrent allocation requests have reached the permissible limit of 1635.
14	The DDNAME requested is unavailable.
18	The DSNAME requested is a member of a concatenated group.
1C	The DDNAME or DSNAME specified is not currently allocated.
20	The requested data set was previously permanently allocated, or was allocated with a disposition of new, and was not deleted. DISP=NEW cannot now be specified.
24	An error occurred in a catalog information routine.
28	The return area you provided for qualifiers was exhausted and more index blocks exist. If you require more qualifiers, provide a larger return area.
30	Reserved.
2C	The previous allocation specified a disposition of DELETE for this non-permanently allocated data set. Request specified OLD, MOD, or SHR with no volume serial number.
34	Request by user denied by installation exit routine.

Chapter 6. Default Service Routine

The default service routine (IKJEHDEF) receives a partially-qualified data set name from a calling routine and constructs a fully-qualified data set name. A fully-qualified data set name has three fields: a user ID, a data set name, and a descriptive qualifier.

In general, if the terminal user refers to a data set without giving a fully-qualified name, IKJEHDEF gets control. The calling routine provides the default routine with the address of the DFPL (default parameter list), which contains the address of the DFPB (default parameter block). The DFPB contains the address of the data set name, as provided by the terminal user.

IKJEHDEF prefixes the user ID to the data set name, checks the data set name against the system catalog, and, if necessary, either inserts the proper qualifier or prompts the user to choose a qualifier.

The default service routine executes below 16 megabytes in virtual storage, in 24-bit addressing mode, but can be invoked by programs executing in either 24-bit or 31-bit addressing mode, as long as all input passed to them is below 16 megabytes. When the default service routine returns control to the program that invoked it, the program continues to execute in the addressing mode (24-bit or 31-bit) that it was in before it passed control to the service routine.

As supplied with TSO, the default service routine resides in SYS1.LPALIB and executes with the protection key of the caller. The default service routine requires about 4,000 bytes of storage.

Method of Operation

Diagram 17 shows how the default service routine constructs a fully qualified data set name.

Default receives control through either a CALL, LINK, or CALLTSSR macro instruction at entry point IKJDFLT in load module IKJEHDEF. At entry, register 1 points to the default parameter list (DFPL).

The default parameter list contains:

- The address of the user profile table (UPT)
- The address of the environment control table (ECT)
- The address of the event control block (ECB)
- The address of the default parameter block (DFPB).

The default parameter block contains a data set name buffer, an entry code, and a control code. The data set name buffer has a 2-byte length field followed by the data set name. If the data set name is less than 44 bytes in length, it must be left-justified and padded on the right with blanks. The entry codes and control codes are set by the calling program to specify the functions required. The following figure describes the entry codes.

Entry Code	Function Requested	Functions Performed by Default
X'00'	Use the qualifier provided by the caller.	Uses qualifier from DFPB, as provided by the caller.
X'04'	Find a qualifier. If there is more than one, prompt the terminal user to choose one.	• Builds a list of possible qualifiers. • Prompts the terminal user to choose one. • Checks his response against the list.
X'08'	Find a descriptive qualifier, but do not interrupt the terminal user.	• Builds a list of possible qualifiers. • Returns control to caller with a control code indicating more than one qualifier was found; thus prompting is necessary.
X'0C'	User qualifier from DFPB or find one from system catalog, or use a new one submitted by the caller.	Does one of the following: • If a qualifier is provided in DFPB, uses it. • If no qualifier is provided: - Builds a list of possible qualifiers. - Sends list to terminal. - Prompts terminal user to choose one from the list or submit a new one.

Note: Entry codes X'80', X'84', X'88', and X'8C' are the same as X'00', X'04', X'08', and and X'0C' respectively except that a catalog name and a password are obtained from the DFPB.

Searching The System Catalog

Default invokes the catalog information routine (IKJEHCIR) to search the system catalog for the required qualifiers. Default must supply the user ID and the data set name as a search argument.

The catalog information routine does the following:

- Uses VSAM catalog management to search the system catalog for the required qualifier.
- Returns a list of qualifiers to default.

Exit From Default

Default returns to the calling control program by issuing a RETURN macro instruction. The routine restores all registers, except register 15, which contains the return code. The possible return codes appear in the "Diagnostic Aids" section of this chapter.

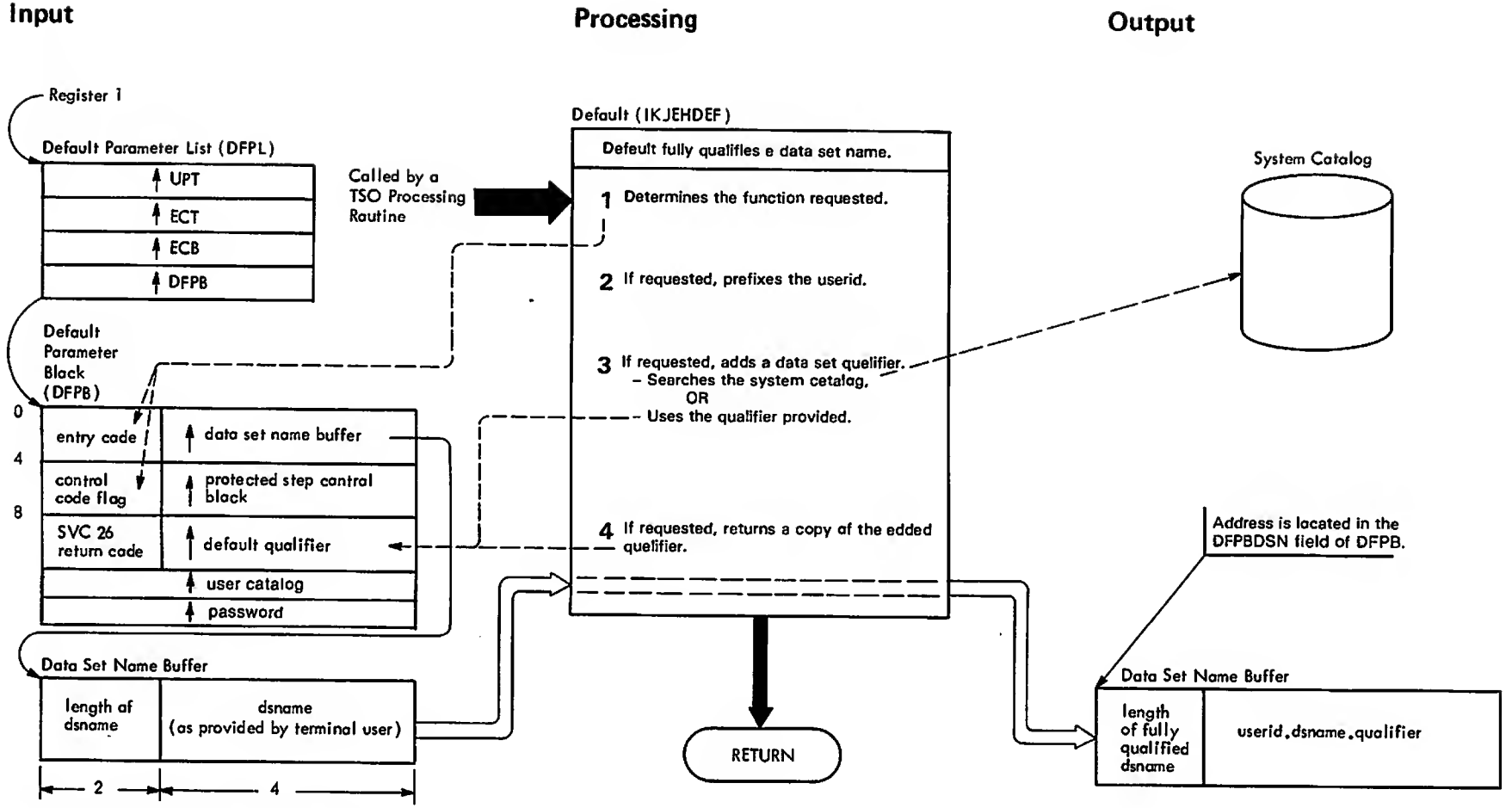


Diagram 17. Default Service Routine (Part 1 of 2)

Key	Description	Routine	Label
1	The default service routine performs one or more functions depending on the entry and control codes passed in the DFPB. The entry and control codes are as follows:	Default Service Routine	IKJEHDEF

Entry Code	Function performed by Default	Control Code Flag	Function performed by Default
X'00'	Uses the qualifier the terminal user provides.	Bit 2	Prefixes the given data set name with the user ID.
X'04'	Searches the system catalog for descriptive qualifiers. If more than one exists, builds a list of possible qualifiers and prompts the terminal user to choose one.	Bit 5	Returns a copy of any added qualifier to the caller.
X'08'	Searches the system catalog for descriptive qualifiers. Attempts to qualify the data set name, but does not interrupt the terminal.	Bit 6 Bit 7	Uses the qualifier the terminal user provides. Issues a message to the terminal user.
X'0C'	Searches the system catalog for the descriptive qualifiers. Accepts the qualifier the terminal user provides; alerts user if it is an old data set.		

Entry codes X'80', X'84', X'88', and X'8C' are the same as X'00', X'04', X'08', and X'0C' respectively, except that a catalog name and a password are obtained from the DFPB.

2	Obtains the userid from the PSCBUSER field of the PSCB and prefixes it to the dsname in the data set name buffer.		ADDUSRID
	The default service routine searches the system catalog by calling the catalog information routine. The catalog information routine then issues a LOCATE macro instruction.		
3	If a data set qualifier is requested, and one is not supplied, the default service routine searches the system catalog by calling the catalog information routine. The catalog information routine in turn issues a LOCATE macro instruction. If a qualifier is supplied, default finds the qualifier by referring to location pointed to by the DFPBQUAL field of the DFPB.		CALLCIR
4	If the calling routine requests a copy of the inserted qualifier, the default service routine inserts this copy in the location pointed to by the DFPBQUAL field of the DFPB.		ADDQUAL

Diagram 17. Default Service Routine (Part 2 of 2)

Program Organization

The default service routine consists of one load module, IKJEHDEF. It resides in SYS1.LPALIB.

Directory

This table contains information to help you find the appropriate program description or assembly listing.

Label	Load Module	Assembly Module	Control Section	Description
ADDNAME	IKJEHDEF	IKJEHDEF	IKJEHDEF	Adds a qualifier to the dsname.
ADDQUAL	IKJEHDEF	IKJEHDEF	IKJEHDEF	Attaches a qualifier to the dsname.
ADDUSRID	IKJEHDEF	IKJEHDEF	IKJEHDEF	Prefixes the userid to the dsname.
BLDLIST	IKJEHDEF	IKJEHDEF	IKJEHDEF	Builds a list of qualifiers.
BADREPLY	IKJEHDEF	IKJEHDEF	IKJEHDEF	Tells the terminal user of any improper replies.
CALLCIR	IKJEHDEF	IKJEHDEF	IKJEHDEF	Calls the catalog information routine to find the data set qualifiers.
CHECKRC	IKJEHDEF	IKJEHDEF	IKJEHDEF	Checks the return code from the catalog information routine.
CLEANUP	IKJEHDEF	IKJEHDEF	IKJEHDEF	Restores registers and frees the core.
CNTRLTST	IKJEHDEF	IKJEHDEF	IKJEHDEF	Checks if a message is required.
COMPARE	IKJEHDEF	IKJEHDEF	IKJEHDEF	Check the terminal user's qualifier against the list.
GETQUAL	IKJEHDEF	IKJEHDEF	IKJEHDEF	Prompts the user for a qualifier.
IKJDFLT	IKJEHDEF	IKJEHDEF	IKJEHDEF	Contains the entry point default.
IKJEHDEF	IKJEHDEF	IKJEHDEF	IKJEHDEF	Serves as a control section.
NOT0C	IKJEHDEF	IKJEHDEF	IKJEHDEF	Checks if a user ID is to be added.
RESPONSE	IKJEHDEF	IKJEHDEF	IKJEHDEF	Check the user's response.
TESTRC	IKJEHDEF	IKJEHDEF	IKJEHDEF	Handles the PUTLINE return code.
TPUT	IKJEHDEF	IKJEHDEF	IKJEHDEF	Writes a line to the terminal.
TPUTGET	IKJEHDEF	IKJEHDEF	IKJEHDEF	Prompts the user to choose a qualifier.

Diagnostic Aids

This section contains default service routine return codes.

Return Code (Hex.)	Meaning of Return Code	Possible Return Code by Entry Code			
		X'00'	X'04'	X'08'	X'0C'
0	Successful operation.	X	X	X	X
4	Unable to obtain qualifier from terminal user. (PUTLINE or PUTGET error).		X		X
8	With qualifiers added, data set length is greater than 44 bytes.	X	X	X	X
C	Permanent I/O error in the system catalog, catalog data set not available, or syntax error in data set name. (The LOCATE return code was X'04', X'14', or X'18')	X	X	X	X
10	Data set exists at some level of index other than the lowest index level specified. (The LOCATE return code was X'10')	X	X	X	X
14	One of the data set names was not found. (LOCATE return code was X'08'.)	X	X	X	X
18	Attention interruption occurred.	X	X	X	X
1C	Invalid parameter: • Invalid entry code, • Data set length not halfword aligned. • Data set length greater than 44 bytes, or • Data set length of 0, except with entry code of X'00'.	X	X	X	X
20	Prompting is necessary to qualify data set name.			X	
24	No qualifiers found. (LOCATE return code was X'0C'.)	X	X	X	X

Chapter 7. Catalog Information Routine

This description of the catalog information routine assumes you are familiar with *Catalog Management Logic.*

The catalog information routine (IKJEHCIR) retrieves information from the system catalog. This information can include a data set name, index name, control volume address, or volume ID. If the user specifies a catalog, IKJEHCIR can request information from the particular catalog. If the user does not specify a catalog, IKJEHCIR uses the system default search and then requests information from the default catalog. An entry code indicates the kind of information requested. Possible information requests include:

- The next level qualifiers for a name.
- All names having the same name as the high-level qualifier and the data set type associated with each name.
- The volume serial numbers and device types associated with a name.

The requester can also ask for combinations of the information above.

The catalog information service routine resides below 16 megabytes in virtual storage and executes in 24-bit addressing mode, but can be invoked by programs executing in either 24-bit or 31-bit addressing mode, as long as all input passed to them is below 16 megabytes. When the catalog information service routine returns control to the program that invoked it, the program continues executing in the addressing mode (24-bit or 31-bit) that it was in before it passes control to the service routine.

The catalog information routine resides in SYS1.LPALIB and executes with the protection key of the caller. The catalog information routine requires about 800 bytes of main storage.

Method of Operation

Diagram 18 shows how the catalog information routine (IKJEHCIR) obtains information from the system catalog.

The catalog information routine receives control through a CALL, LINK, or CALLTSSR macro instruction at entry point IKJEHCIR. At entry, register 1 points to the catalog information routine parameter list (CIRPARM). CIRPARM contains:

- The option code requesting a particular service.
- The address of the search argument. This search argument consists of a user ID and a data set name; both are names of catalog index levels.
- The address of the user catalog name.
- The address of the work area; the calling program supplies this area.
- The address of the save area; the calling program supplies this area.
- The address of the data set or catalog password.

The catalog information routine returns to the calling program using a RETURN macro instruction. All registers except 15 are restored. At exit, register 15 contains a return code.

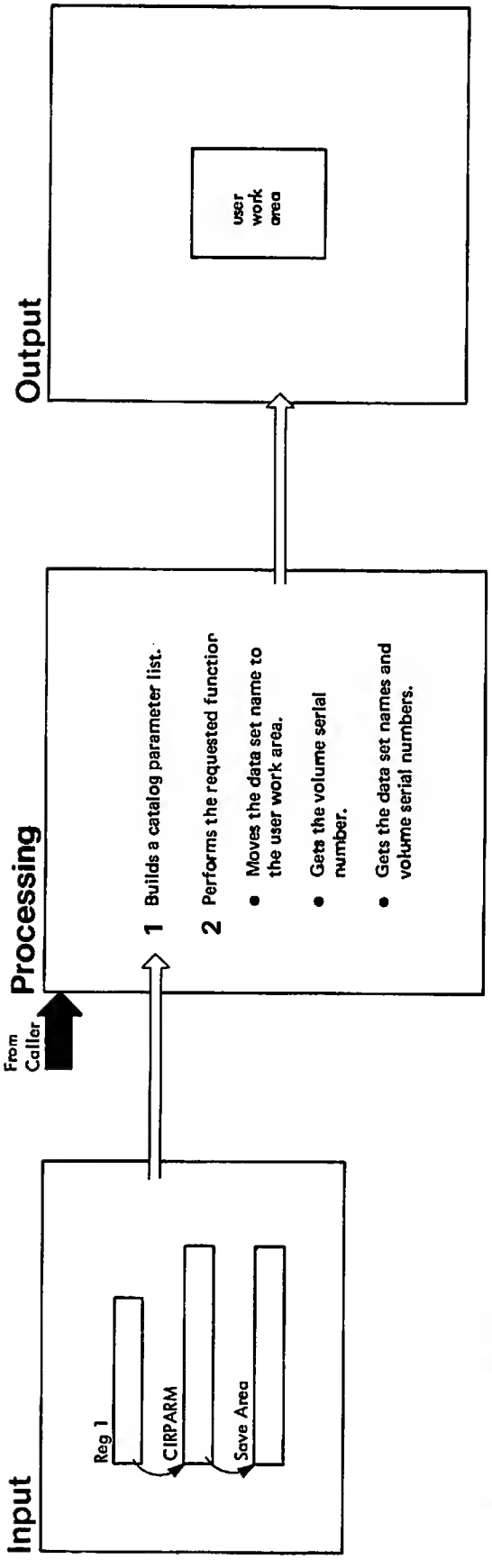


Diagram 18. Catalog Information Routine (Part 1 of 2)

Key Description		Routine
1 Builds a catalog parameter list from the information in CIRPARM. The entry code in CIRPARM determines the function to be performed.		IKJEHCIR
2 Uses SVC 26 to perform the function requested by the entry code:		
X'01'	Moves the data set names having one more level of qualifier above what the user specified to the user's work area.	
X'02'	Moves all data set names to the user's work area.	
X'04'	Gets a volume associated with a given data set name.	
X'05'	Gets the next level data set names and volume information.	
X'06'	Gets all level data set names and volume information.	
Returns control to the caller.		

Diagram 18. Catalog Information Routine (Part 2 of 2)

Program Organization

This section describes the organization of the catalog information routine.

The catalog information routine consists of one load module, IKJEHCIR. As supplied with TSO/E, the catalog information routine resides in SYS1.LPALIB.

Directory

This table contains information to help you find the appropriate program description or assembly listing. It correlates information from three sources:

- The source code
- The executable load modules
- This manual

Label	Load Module	Assembly Module	Control Section	Description
IKJEHCIR	IKJEHCIR	IKJEHCIR	IKJEHCIR	Contains the control section name and program entry point.
ENTRY01	IKJEHCIR	IKJEHCIR	IKJEHCIR	For a given name, processes the list of qualifiers.
ENTRY02	IKJEHCIR	IKJEHCIR	IKJEHCIR	Processes a list of data set names starting with a given level.
ENTRY04	IKJEHCIR	IKJEHCIR	IKJEHCIR	Processes a list of volumes.
ENTRY05	IKJEHCIR	IKJEHCIR	IKJEHCIR	Processes a list of qualifiers and volumes.
ENTRY06	IKJEHCIR	IKJEHCIR	IKJEHCIR	Processes a list of data set names and volumes.

Diagnostic Aids

This section contains the catalog information routine and LOCATE macro instruction return codes.

Catalog Information Routine

Return Code Hexadecimal	Meaning
0	Successful completion of the request.
4	The LOCATE macro instruction failed. The CIRLOCRC field stores the LOCATE return code.
0C	LOCATE returned volumes, indicating a dsname (fully qualified) was passed in the parameter list, but options other than volumes were requested. The list of the volumes returned by LOCATE is in the work area.

Locate

Return Code Hexadecimal	Meaning
0	Successful completion of the request.
4	Required VSAM volume was not mounted or the specified volume was not open.
8	The data set name qualifier was not found.
18	A permanent I/O error was found when processing the catalog.
20	User work outside user region or invalid user supplied parameter list.
24	User catalog must be allocated and opened.
2C	Work area too small.
38	Password verification failure.
3C	STEPCAT or JOBCAT not open.

Chapter 8. DAIR/SVC99 Error Code Analyzer

The DAIR/SVC99 error code analyzer (IKJEFF18 -- DAIRFAIL) examines SVC99 (dynamic allocation) or DAIR failure codes and issues the appropriate informational message. This processing provides the user, at his option, meaningful diagnostic information about failures that occur during these functions. DAIRFAIL can also issue the message and/or return the requested informational message to the caller in buffers supplied by the calling program. This process of returning the requested message is called "extracting the message."

The DAIRFAIL service routine resides below 16 megabytes in virtual storage and executes in either 24-bit or 31-bit addressing mode. If in 31-bit addressing mode, DAIRFAIL can process input above 16 megabytes.

Method of Operation

Diagram 19 shows how DAIRFAIL analyzes errors and processes the messages as requested. Because DAIRFAIL can access the return codes that dynamic allocation, DAIR, or the catalog management routines return, DAIRFAIL can determine the cause of the failure.

The following list briefly describes the steps in which DAIRFAIL processes.

- DAIRFAIL examines the caller identifier to determine the type of error to be processed. Possible error types include:
  - Errors from the SVC 99 interface to dynamic allocation
  - Errors from the DAIR interface to dynamic allocation.
- DAIRFAIL takes information (DSNAME, DDNAME, VOLSER, etc.) from the text units for SVC 99 errors and the DAIR parameter block for DAIR errors. DAIRFAIL then uses this information as text in an error message.
- DAIRFAIL examines the applicable return code from dynamic allocation, DAIR, or the catalog management routines to select the proper informational message.
- DAIRFAIL prepares a parameter list that contains:
  - The message identifier
  - The variables for insertion into the message
  - The address of the message control section
  - If message extraction is requested, the address of the return buffers and their lengths.

  DAIRFAIL passes control to TSO/E's general purpose message issuer (IKJEFF02) for completion, and processes the message as requested.
- DAIRFAIL terminates by setting the standard return code register to zero and returning to the calling command processor.

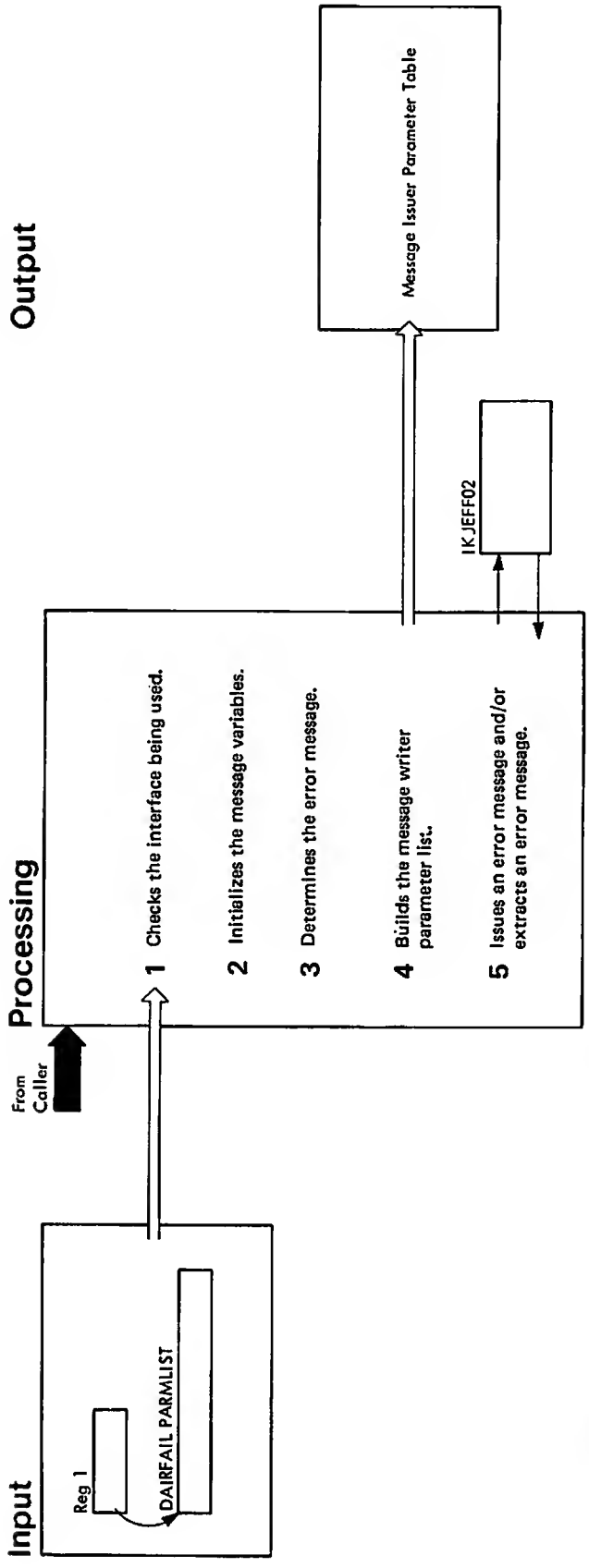


Diagram 19. Analyzing DAIR/SVC99 Error Codes (Part 1 of 2)

Key	Description	Routine
1	Checks for a dynamic allocation or DAIR error interface.	IKJEFF18
2	Initializes the message variables (data set name, file name, VOLSER, etc.). For the dynamic allocation error interface, uses the SVC99 parameter list to initialize the variables. For the DAIR error interface, uses the DAIR parameter block to initialize the variables.	
3	Compares the dynamic allocation error code to a table of error codes. The error code table specifies the identifier of the error message to be issued.	
4	Builds a message writer parameter table with the selected message identifier and variables to be used in the message text.	
5	Uses IKJEFF02 to issue the error message to the terminal user and/or extract the error message. Returns control to the caller.	

Diagram 19. Analyzing DAIR/SVC99 Error Codes (Part 2 of 2)

Program Organization

This section describes the organization of the DAIR/SVC99 error code analyzer (DAIRFAIL).

Program Hierarchy

Users can invoke DAIRFAIL to get a diagnostic message related to a dynamic allocation failure. DAIRFAIL consists of the following routines:

IKJEFF18 - the DAIRFAIL load module, which contains one CSECT (IKJEFF18) to control and execute the analysis of error return codes, and another CSECT (TSMSG) that contains the pertinent message segments for the construction of meaningful diagnostic messages.

IKJEFF02 - TSO/E's general purpose message issuer, which formats, issues, and/or extracts the diagnostic messages under the control of IKJEFF18.

Module Operation

The following descriptions outline the processing operations in the DAIR error code analyzer routines.

IKJEFF18 -- DAIR/SVC99 Error Code Analyzer (DAIRFAIL)

- Examines error return codes from SVC99 (dynamic allocation) or DAIR.
- Constructs an appropriate informational message based on the error code.
- Invokes TSO/E's general purpose message issuer to issue the message to the terminal by a PUTLINE write-to-programmer macro instruction, or by issuing and/or extracting the message.

Directory

This table contains information to help you find the appropriate module operation description or assembly listing.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
IKJEFF18	DAIR/SVC99 Error Code Analyzer (DAIRFAIL)	IKJEFF18	IKJEFF18	IKJEFF18	Analyzes error codes and prepares appropriate informational messages.
TSMSG	DAIRFAIL Message CSECT	IKJEFF18	IKJEFF18	IKJEFF18	Contains DAIRFAIL message frameworks.
IKJEFF02	TSO Message Issuer	IKJTSLAR	IKJTSLAR	IKJEFF02	Issues and/or extracts the DAIRFAIL messages.

Chapter 9. TSO/E Message Issuer (IKJEFF02)

The TSO/E message issuer sends an error message to the terminal user (and/or extracts messages) using the following steps:

1. Selects the message requested.
2. Converts values to printable hexadecimal or decimal.
3. Inserts variable text.
4. Compresses blanks.
5. Strips off message IDs (at the user's request).

IKJEFF02 can then either (1) issue the message(s) to the user at the terminal and/or (2) put the message(s) in buffers the caller provides. The second alternative is called extracting the message.

The TSO/E message issuer service routine resides above 16 megabytes in virtual storage and executes in 31-bit addressing mode via the TSO/E service linkage assist routine. It can process input above or below 16 megabytes in virtual storage.

When a program invokes the TSO/E message issuer, the TSO/E service linkage assist routine receives control. This routine resides below 16 megabytes in virtual storage and executes in the addressing mode in which it was invoked. It branches to the service routine using the BASSM instruction, which switches to 31-bit addressing mode. The service routine branches back to the TSO/E service linkage assist routine using the BASM instruction, which restores the original addressing mode. The TSO/E service linkage assist routine then restores the registers and returns control to the program.

Method of Operation

Diagram 20 shows how the message issuer builds the message and issues or extracts the message.

The message issuer:

- Selects the message from the message module specified by the caller.
- Examines the options requested by the caller and builds the message accordingly.
- Issues and/or extracts the message to the user at the terminal - issues a write-to-operator or a write-to-programmer macro instruction.

Diagram 20. TSO Message Issuer (Part 1 of 2)

Key	Description	Routine
1	Checks the MSGTABLE parameter list for the requested options. If parameter problems exist, issues an error message.	IKJEF02R
2	Searches the message control section (CSECT) for a message, using the message ID passed in the parameter list.	MSGSRCH
3	Builds the message according to the options requested by the caller.	BLDLOOP
4	As requested by the options, either: • Issues the message as a: - Write-to-operator - Write-to-programmer - PUTLINE - PUTGET • Issues and/or extracts the messages according to the options requested by the caller.	WTPMSG

Diagram 20. TSO Message Issuer (Part 2 of 2)

Program Organization

This section describes the organization of the TSO/E message issuer.

Program Hierarchy

The message issuer can issue a message to the terminal user or issue and/or extract a message. The message issuer consists of the following routine:

IKJEFF02 - The message issuer load module contains one CSECT (IKJEF02R) to control and execute the message processing and another CSECT (IKJEFF2M) to contain error messages issued by IKJEF02R.

Module Operation

The following describes the processing operations in the message issuer routines.

IKJEFF02 -- TSO/E Message Issuer

- Searches for the message requested in the message control section passed by the caller.
- Builds the message based on the options specified.
- Issues the message to the user's terminal or issues and extracts the message.

Directory

This table contains information to help you find the appropriate module operation description or assembly listing.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
IKJEF02R	TSO Message Issuer	IKJEF02R	IKJEF02R	IKJEF02R	Issues and/or extracts messages.
IKJEFF2M	TSO Message Issuer Error Message CSECT	IKJEF02R	IKJEF02R	IKJEFF2M	Contains TSO message issuer message text.

Chapter 1. Introduction

TSO/E extends the capabilities of the operating system to include general-purpose time sharing for terminal users.

TSO/E System

TSO/Executes as an authorized program under the control of the operating system control program. A terminal user describes the work to be processed by entering commands at the terminal. The terminal monitor program (TMP) receives these commands and sets them up for processing.

The TMP consists of three task level routines: the TMP initialization routine (IKJEFT01), the TMP mainline routine (IKJEFT02), and the TMP second-level routine (IKJEFT09). The TMP mainline routine executes in supervisor state or problem program mode. The TMP second-level routine **always** executes in problem program mode.

Two TSO/E commands execute on the same task level as the TMP mainline: the TIME and CALL commands. The TEST command executes on the same task level as the TMP second-level routine (which is a subtask of the TMP mainline routine).

Any command processor whose name appears in the APF command table (module IKJEFTE2) or the APF program table (module IKJEFTE8) receives control as an APF-authorized subtask, via a parallel TMP mainline routine (IKJEFT02) or the CALL command processor. The TMP second-level routine or the CALL command processor attaches all other commands and programs.

Note: The APF command tables can be initialized and maintained using members of the PARMLIB data set. For information about using the PARMLIB data set, see *TSO Extensions Customization.*

As the TMP and the command processors execute, they can invoke the TSO/E service routines to perform the following operations:

- Handling input/output operations to or from terminals.
- Searching input buffers for TSO/E commands and command parameters.
- Allocating and freeing data sets and performing other data management functions.

All service routines execute at the same task level as the program that invokes them.

The IBM-supplied terminal monitor program and service routines reside above 16 megabytes in virtual storage. The terminal monitor program executes in 31-bit addressing mode, except when interfacing with a routine that must receive control in 24-bit mode. The service routines, however, fall into three categories:

- Service routines that can only be invoked in 24-bit addressing mode.
- Service routines that can be invoked in either 24-bit or 31-bit addressing mode, but must receive all input below 16 megabytes in virtual storage.

- Service routines that can be invoked in either 24-bit or 31-bit addressing mode, and can receive input from below or above 16 megabytes in virtual storage. These routines can only process input above 16 megabytes when they are invoked in 31-bit addressing mode.

MVS/Extended Architecture Considerations

The CLIST attention facility routines (IKJCAF and IKJCAFR) require that MVS/System Product Version 2 Release 2.0 be installed on the system.

IKJCAF processes a CLIST attention exit and establishes its own recovery routine. IKJCAFR processes any ABEND that occurs under the mainline. Both routines run above 16 megabytes in virtual storage, in the key and state of the calling program or function. See Chapter 12, "CLIST Attention Facility", for more information about these routines.

The following service routines were modified for MVS/XA:

- (IKJEHCIR) catalog information routine
- (IKJEHDEF) default service routine.

Although these service routines execute below 16 megabytes in virtual storage, in 24-bit addressing mode, they can be invoked by programs executing in either 24-bit or 31-bit addressing mode as long as all input passed to them is below 16 megabytes. When the service routines are invoked, they save the addressing mode of the program that invokes them. When the service routine returns control to the invoking program, that program restores its original addressing mode (24-bit or 31-bit).

TSO Extensions Considerations

The following service routines reside above 16 megabytes in virtual storage, and execute in 31-bit addressing mode, but can be invoked by programs executing in either 24-bit or 31-bit addressing mode. Those programs executing in 24-bit addressing mode can use a linkage assist routine in order to use the service routines. These routines can process input passed to them above or below 16 megabytes in virtual storage. The list source descriptor (LSD), which is passed to the STACK routine, must reside below 16 megabytes in virtual storage.

- CLIST variable access routine
- TSO/E message issuer
- STACK routine
- GETLINE routine
- PUTLINE routine
- PUTGET routine
- Parse routine
- Scan routine.

When a program invokes one of these services, the TSO/E service linkage assist routine (LAR) receives control. Linkage assist routines are programs that handle the mode switching needed to pass control between certain programs operating in 24-bit and 31-bit addressing modes. The TSO/E service LAR resides below 16 megabytes in virtual storage and executes in the addressing mode in which it is invoked. It branches to the service routine using the BASSM instruction, which, if necessary, switches to 31-bit addressing mode. The service routine branches back to the TSO/E service LAR using the BSM instruction, which restores the original addressing mode. The TSO/E service LAR then restores the registers and returns control to the program.

The following service routines reside below 16 megabytes in virtual storage, but execute in either 24-bit or 31-bit addressing mode.

- (IKJDAIR) dynamic allocation interface routine (DAIR)
- (IKJEFF18) DAIR/SVC99 error analyzer routine (DAIRFAIL)
- (IKJEFTSR) TSO/E service routine.

If invoked in 31-bit addressing mode, these routines can process input above 16 megabytes in virtual storage.

Terminal Monitor Program

The terminal monitor program obtains TSO/E commands, gives control to TSO/E command processors, and monitors their execution.

The IBM-supplied TMP is a program executed via the user logon procedure or as a background job step. An installation can write a program similar to the TMP and substitute it for the IBM-supplied TMP for local application.

The TMP does the following:

- Issues broadcast messages during LOGON processing (foreground only).
- Obtains a new command and gives control to the appropriate command processor.
- Handles attention requests.
- Initiates recovery from errors in a command processor/program or one of its subtasks.
- Initiates recovery from errors in its own routines.
- Returns control to the LOGON/LOGOFF scheduler when the operator issues a STOP command or when the terminal user enters a LOGON or LOGOFF command.

TSO/E Command Processors and User Programs

TSO/E command processors are programs that perform the operations requested by TSO/E commands such as EDIT, CALL, RUN, and ALLOCATE. An installation can write similar programs for use as command processors as described in *TSO Extensions Programming Guide*.

Some TSO/E command processors call on standard system processors to perform the requested function. For example, the COBOL command processor sets up a standard calling sequence according to the options selected by the user, then transfers control to the ANS COBOL compiler to compile the user's program. Except for the special formatting of output and messages, the compiler operates exactly as it would in a non-TSO environment.

TSO/E Service Routines

The TMP, TEST, and other TSO/E command processors use the TSO/E service routines. In general, they perform services needed by all TSO/E problem programs. Their use as subroutines saves repetitive coding in the command processors.

Terminal I/O Service Routines

The terminal I/O service routines handle terminal input/output operations required by the LOGON/LOGOFF scheduler, the TMP, the TSO/E command processors, and other TSO/E programs.

There are four terminal I/O service routines:

- STACK
- GETLINE
- PUTLINE
- PUTGET

Command SCAN and PARSE Service Routines

Command scan and parse search the command buffer for TSO/E commands and their parameters. In general, the TMP invokes command scan. However, the TEST command processor and TSO/E command processors that accept subcommands can also invoke command scan. The TSO/E command processors usually invoke parse.

Dynamic Allocation Interface Routine

The dynamic allocation interface routine (DAIR) allocates and frees the data sets needed by the TSO/E command processors and other TSO/E programs. In general, DAIR obtains information about a data set and, if necessary, invokes dynamic allocation routines to perform the requested function.

Default Service Routine and Catalog Information Routine

The default service routine constructs a data set name that follows TSO/E data set naming conventions. The catalog information routine obtains information from the system catalog.

DAIR Error Code Analyzer

The DAIR error code analyzer (DAIRFAIL) examines dynamic allocation, DAIR, or catalog management failure codes and issues the appropriate informational message.

TSO/E Service Routine

The TSO/E service routine (IKJEFTSR or TSOLNK) allows a TSO/E user to invoke any command, program, or CLIST from an authorized or unauthorized environment. Any unauthorized program or command processor that uses the TSO/E service routine can ignore the consideration of authorized or unauthorized environments and programs. However, an authorized program or command processor can use the TSO/E service routine to invoke only authorized programs, command processors, or CLISTs consisting of only authorized command processors and programs. A command or program can invoke the TSO/E service routine in both foreground and background TSO/E sessions.

Chapter 10. TSO Service Facility

The TSO/E service facility consists of the following routines:

- TSO/E service routine (IKJEFTSR)
- TSO/E service routine SVC (IGX00035)
- TSO/E parameter verification routine (IKJEFTPV).

IKJEFTSR invokes IGX00035; both modules return information to the caller. IGX00035 allows its caller to request the TMP to invoke any program, TSO/E command processor (CP), or CLIST. Although IKJEFTSR can be used to invoke an unauthorized program or command processor from within an unauthorized environment, it is most useful for invoking authorized programs or CPs from within an unauthorized environment. Attempting to invoke an unauthorized program from an authorized environment results in a return code.

When invoking a program or CP via the TSO/E service routine, the caller puts the name of the program or CP in a parameter list. The caller of IKJEFTSR specifies the address of the parameter list, and IKJEFTSR passes the address to the SVC (IGX00035).

Note: For additional information concerning IKJEFTSR and its parameter list, see *TSO Extensions Programming Services.*

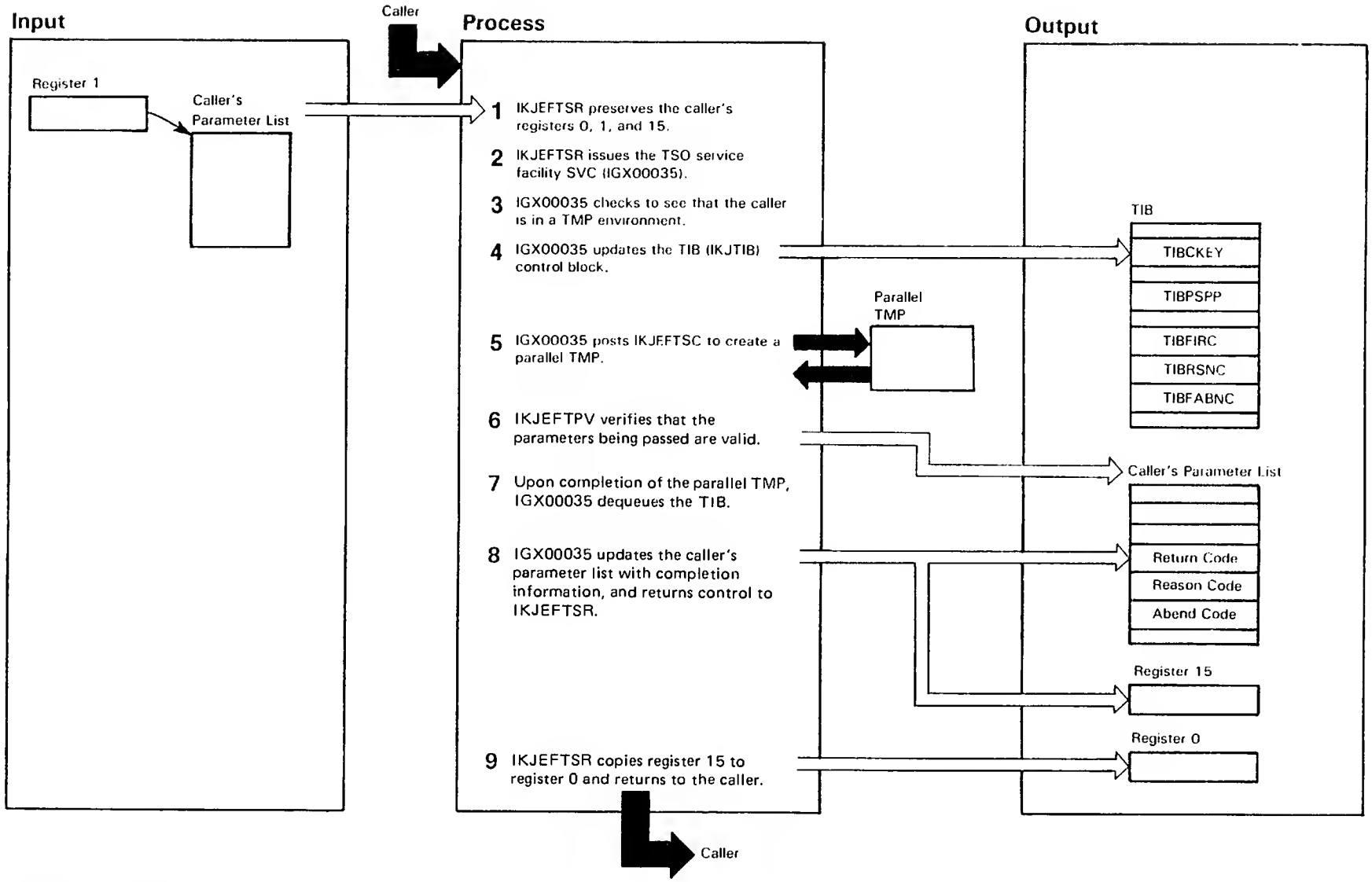


Diagram 21. TSO Service Facility (Part 1 of 2)

Key Description	Routine
1. IKJEFTSR preserves register 1, which contains the address of the parameter list passed by the invoker.	IKJEFTSR
2. IKJEFTSR issues IGX00035 and passes it the address of the parameter list in register 1.	IKJEFTSR
3. IGX00035 verifies that the caller is in a TMP environment.	IGX00035
4. IGX00035: • Updates the TIB control block with the: – Address of the caller's parameter list (TIBPSPP) – Caller's key (TIBCKEY) • Sets the following fields to X'FFFFFFFF': - Function return code (TIBFRC) - Reason code (TIBRSCN) - Function abend code (TIBFABNC) • Places the TIB on the queue for processing by a parallel TMP.	IGX00035
5. IGX00035 posts IKJEFTSC to establish a parallel TMP and waits for the requests to be completed. (See Diagram 4 for a description of parallel TMP processing.)	ICX00035
6. IKJEFTPV verifies that the parameters being passed are valid.	IKJEFTPV
7. Upon completion of the request, IGX00035 dequeues the TIB control block.	IGX00035
8. IGX00035 returns the following completion information in the locations pointed to by the 4th, 5th, and 6th fields of the caller's parameter list. It gets that information from the TIB control block fields shown in parentheses. • Function return code from the program or CP (ITBFRC) • Reason code from the program or CP (TIBRSNC) • Function abend code (TIBFABNC). It copies the TSO/E service facility return code from the TIBRC Field of the TIB control block to register 15 and then passes control to IKJEFTSR.	IGX00035
9. IKJEFTSR preserves register 15, copies information from register 15 into register 0, and returns to the caller. Registers 15 and 0 contain the TSO/E service facility return code.	IKJEFTSR

Diagram 21. TSO Service Facility (Part 2 of 2)

Program Organization

This section describes the organization of the TSO/E service facility.

Program Hierarchy

The TSO/E service facility consists of the following load modules:

IKJEFTSR - The TSO/E Service Routine invokes IGX00035 and passes a return code to its caller.

IGX00035 - The TSO/E Service Routine SVC causes the invoked program or CP to execute under a parallel TMP with appropriate authorization and passes return information to the caller.

Note: The TSO/E service facility also contains IKJEFTPV, the TSO/E parameter verification routine that ensures parameters passed to IKJEFTSR are in storage and available to the caller. Load module IKJEFT01 contains IKJEFTPV.

Module Operation

The following descriptions outline the processing operations of the TSO/E service routines:

IKJEFTSR -- TSO/E Service Routine

This routine allows its caller to request the TMP to attach any program or TSO/E command through SVC 109. IKJEFTSR:

- Preserves registers 0, 1, and 15 to pass a parameter list and return codes.
- Issues a parallel processing request via SVC 109 with a routing code of 23 (hex) in register 15.
- Passes the parallel processing return code to the caller in registers 0 and 15. (High level languages require the return code in register 0.)

IGX00035 -- TSO/E Service Routine SVC

This SVC allows any caller to request the TMP to attach any program or TSO/E command. The program or TSO/E command is specified by the caller of this SVC in the input parameter list. IGX00035:

- Establishes recovery.
- Sets TIBVERIP to indicate that IKJEFTSC must call IKJEFTPV for parameter verification.
- Verifies that the parameters are valid if the caller of the SVC was AMODE 24.
- Verifies that the SVC was called from a TMP environment (foreground or background).
- Place the following information in a communication area (TIB):
  - Address of the parameter list passed by the caller of the SVC.
  - User's key.
- Queues the TIB for processing by the TMP.
- Posts the TMP to request establishment of a parallel TCB structure to process the request.
- Waits for completion of the processing of the request.
- Dequeues the TIB.
- Passes return information to the caller of the SVC:
  - TMP processing return code.
  - Function return code.
  - Function reason code.
  - Function ABEND code.
  - Deletes recovery.

IGX00035 Recovery Operation

The recovery routine performs the following processing:

- Sets a return code to indicate the type of failure.
- Dequeues the TIB, if appropriate.
- If this error is not an ABEND recursion, then retries.
- Otherwise, continues with termination

IKJEFTPV -- Parameter Verification Routine

This routine validates the parameters passed by the caller of the TSO/E service facility (TSF). It verifies that the parameters are in storage that is accessible to the caller of the TSO/E service facility. IKJEFTPV:

- Determines that the number of parameters is either six or seven.
- Verifies the caller's authority to read the parameters.
- Verifies that all flags have valid values and that reserved fields are zero.
- Verifies that the TSF function buffer is less than 32K-4.
- Verifies the caller's authority to write in the return fields.
- Verifies the caller's authority to read the program parameters if the seventh (program) parameter is present.
- Passes return information to the caller:
  - Validation return code.
  - TSF return code.
  - TSF reason code.

Directory

This table contains information to help you find the appropriate module operation description or assembly listing.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
IKJEFTSR	TSO Service Routine	IKJEFTSR	IKJEFTSR	IKJEFTSR	Invokes the TSO service facility SVC.
TSOLNK	TSO Service Routine	IKJEFTSR	IKJEFTSR	IKJEFTSR	Alias for the TSO service routine.
IGX00035	TSO Service Routine SVC	IGX00035	IGX00035	IGX00035	Requests the TMP to attach a program or CP to run under a parallel TMP.

Diagnostic Aids

This section contains a summary of the return codes, and their meanings.

Routine	Hexadecimal Return Code	Meaning
IKJEFTSR	none	Passes return code from parallel processing SVC.
IGX00035	00	Processing successful.
	10	The parameter list is not accessible: - Contains addresses of data in protected storage - Parameter list is incomplete.
	14	The parameter list to IKJEFTSR contains an error or the environment is incorrect.
	18	The TSO routines associated with the TSO service facility encountered an unexpected failure.
	1C	The parameter list of an AMODE 24 caller contains 31-bit addresses.
IKJEFTPV	00	The caller's parameters are valid.
	04	The caller's parameters are not valid.

Chapter 11. TSO/E Service Linkage Assist Routine

The TSO/E service linkage assist routine (IKJTSLAR) is an interface to the I/O, CLIST variable, message issuer, command scan, and parse service routines. These TSO/E service routines execute in 31-bit addressing mode and reside above 16 megabytes in virtual storage. The TSO/E service linkage assist routine resides above 16 megabytes and can be invoked in either 24-bit or 31-bit addressing mode. It routine ensures that the service routines are invoked in 31-bit addressing mode.

When a program invokes a service, the TSO/E service linkage assist routine gets control, cleans up the parameter list address, switches addressing modes (if necessary), and passes control to the desired service. After the service routine executes, the TSO/E service linkage assist routine receives control, restores the caller's registers, and returns to the caller.

Method of Operation

This section describes the method of operation for the TSO/E service linkage assist routine.

The TSO/E service linkage assist routine has an entry point for each service routine to which it can pass control. The entry points are:

Entry Point In IKJTSLAR	Entry Point Name of Service Routine
IKJSCAN	IKJSCANR
IKJPARS	IKJPARSR
IKJEFF02	IKJEF02R
IKJPUTL	IKJPUTLR
IKJPTGT	IKJPTGTR
IKJGETL	IKJGETLR
IKJSTCK	IKJSTCKR
IKJCT441	IKJT441R

Diagram 22 shows how the TSO/E service linkage assist routine branches to the service routines residing above 16 megabytes in virtual storage.

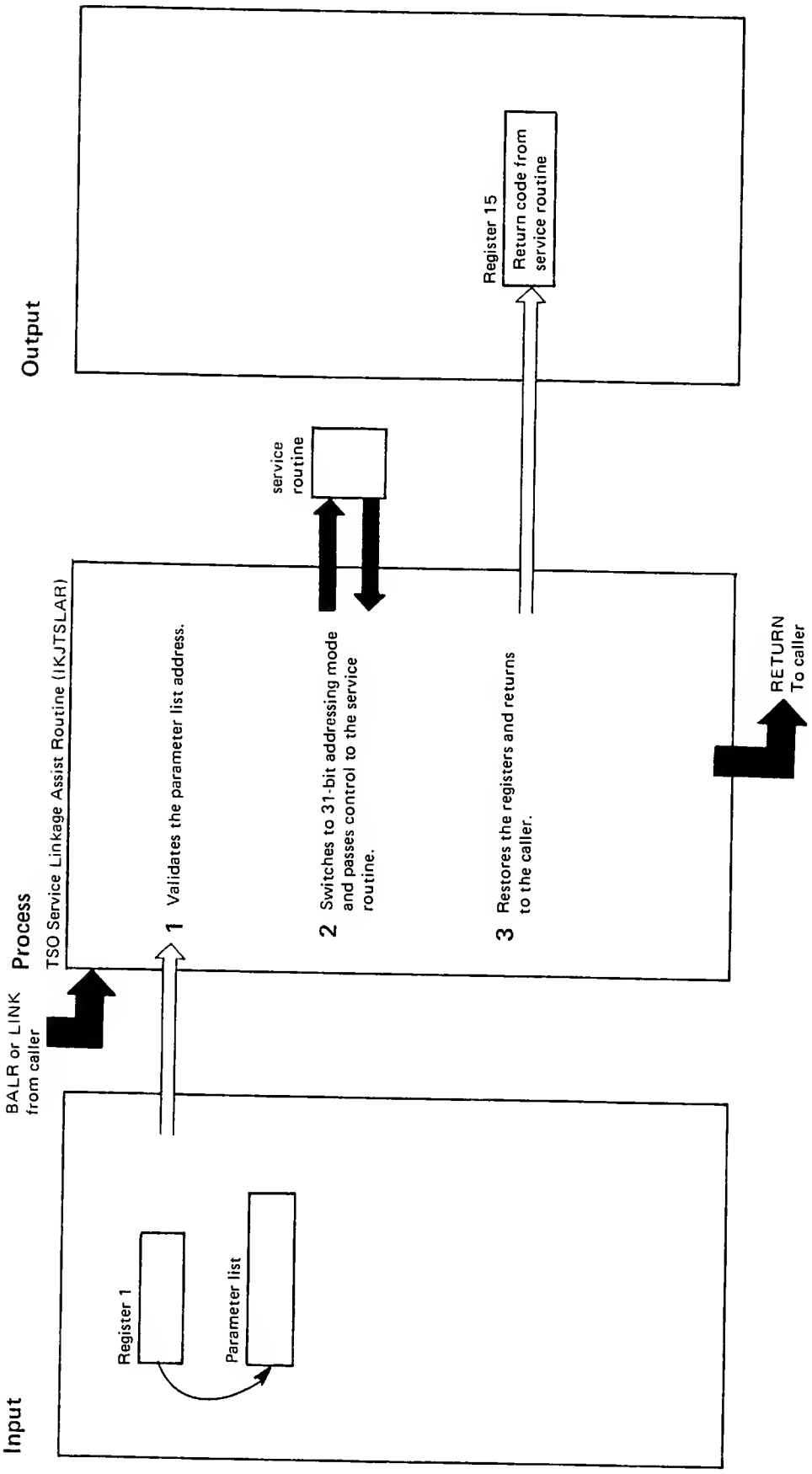


Diagram 22. TSO Service Linkage Assist Routine (IKJTSLAR) (Part 1 of 2)

Key Description	Routine
1 If the caller is running in 24-bit addressing mode, sets the high order byte of the parameter list address (register 1) to zero, to make certain the address will be valid in 31-bit mode. 2 If necessary, switches to 31-bit addressing mode, using a BASSM instruction to branch to the service routine. The routine that receives control depends on the entry point at which IKJTSLAR receives control. (The list of entry points at which IKJTSLAR can receive control appears in the "Method of Operation" section of this chapter.) 3 After the service routine returns to IKJTSLAR, restores the caller's registers and returns to the caller with the return code from the service routine in register 15.	IKJTSLAR

Diagram 22. TSO Service Routine Linkage Assist Routine (IKJTSLAR) (Part 2 of 2)

Program Organization

This section describes the organization of the TSO/E service linkage assist routine (IKJTSLAR).

Program Hierarchy

The TSO/E service linkage assist routine enables a program residing below 16 megabytes in virtual storage to link to a service routine above 16 megabytes. The routine is organized as follows:

IKJTSLAR - The TSO/E service linkage assist load module contains one CSECT (IKJTSLAR) to control common processing for the different service routines. There is one entry point for each TSO/E service routine to which the linkage assist routine can pass control.

Module Operation

The following descriptions outline the processing operations of the TSO/E service linkage assist routine.

IKJTSLAR -- TSO/E Service Linkage Assist Routine

- If running in 24-bit addressing mode, sets the high order byte of the parameter list address to zero to make certain that the address will be valid in 31-bit addressing mode.
- If necessary, switches to 31-bit addressing mode and passes control to the desired service routine using a BASSM instruction.
- On receiving control from the service routine, restores the registers and passes the return code from the service routine to the caller in register 15.

Directory

This table contains information to help you find the appropriate module operation description or assembly listing.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
IKJTSLAR	TSO Service Linkage Assist Routine	IKJTSLAR	IKJTSLAR	IKJTSLAR	Provides linkage to TSO service routines running above 16 megabytes in virtual storage.
IKJCT441	CLIST Variable Access Routine	IKJTSLAR	IKJTSLAR	IKJTSLAR	Retrieves and updates the CLIST variables.
IKJEFF02	TSO Message Issuer Routine	IKJTSLAR	IKJTSLAR	IKJTSLAR	Issues and/or extracts the DAIRFAIL messages
IKJGETL	GETLINE Service Routine	IKJTSLAR	IKJTSLAR	IKJTSLAR	Gets a line of input from the terminal or from the current source of input.
IKJPARS	PARSE Service Routine	IKJTSLAR	IKJTSLAR	IKJTSLAR	Obtains storage for PWORK and RWORK.
IKJPTGT	PUTGET Service Routine	IKJTSLAR	IKJTSLAR	IKJTSLAR	Obtains commands and operands.
IKJPUTL	PUTLINE Service Routine	IKJTSLAR	IKJTSLAR	IKJTSLAR	Formats and puts lines to the terminal or output data set.
IKJSCAN	SCAN Service Routine	IKJTSLAR	IKJTSLAR	IKJTSLAR	Scans buffer for the command name.
IKJSTCK	STACK Service Routine	IKJTSLAR	IKJTSLAR	IKJTSLAR	Manages the input stack which determines the current source of input.

Diagnostic Aids

This section contains a summary of the ABEND and return codes and their meanings.

Routine	Return Code Hexadecimal	Meaning
IKJTSLAR	0	Successful completion.
	8	Invalid entry point. (This is a reason code.)

Chapter 12. CLIST Attention Facility

The CLIST attention facility is a set of routines that any program or function can call to process a CLIST with a CLIST attention exit. The CLIST attention facility consists of the following routines:

- CLIST attention facility mainline routine (IKJCAF)
- CLIST attention facility recovery routine (IKJCAFR).

IKJCAF processes a CLIST attention exit and establishes its own recovery routine. IKJCAFR processes any ABEND that occurs under the mainline. Both routines run above 16 megabytes in virtual storage, in the key and state of the calling program or function.

Note: For additional information about the CLIST attention facility and its parameter list, see *TSO Extensions Programming Services*.

Method of Operation

Diagram 23 shows how the CLIST attention facility mainline routine (IKJCAF) processes a CLIST attention exit. The CLIST attention facility receives control at entry point IKJCAF via a CALLTSSR macro instruction. Upon entry, register 1 points to the CLIST attention facility parameter list (IKJCAFPL).

IKJCAFPL contains:

- Address of the terminal attention interrupt element (TAIE)
- Address of the input/output parameter list (IOPL)
- Address of the PUTGET parameter block (PGBP)
- Address of the STACK parameter block (STPB)
- ABEND code storage area
- ABEND reason code storage area.

If a failure occurs while IKJCAF is processing a CLIST attention exit, IKJCAF passes diagnostic information to the caller via IKJCAFPL.

Output from the CLIST attention facility consists of a return code and/or an input buffer that contains the TSO/E command coded in the CLIST attention exit.

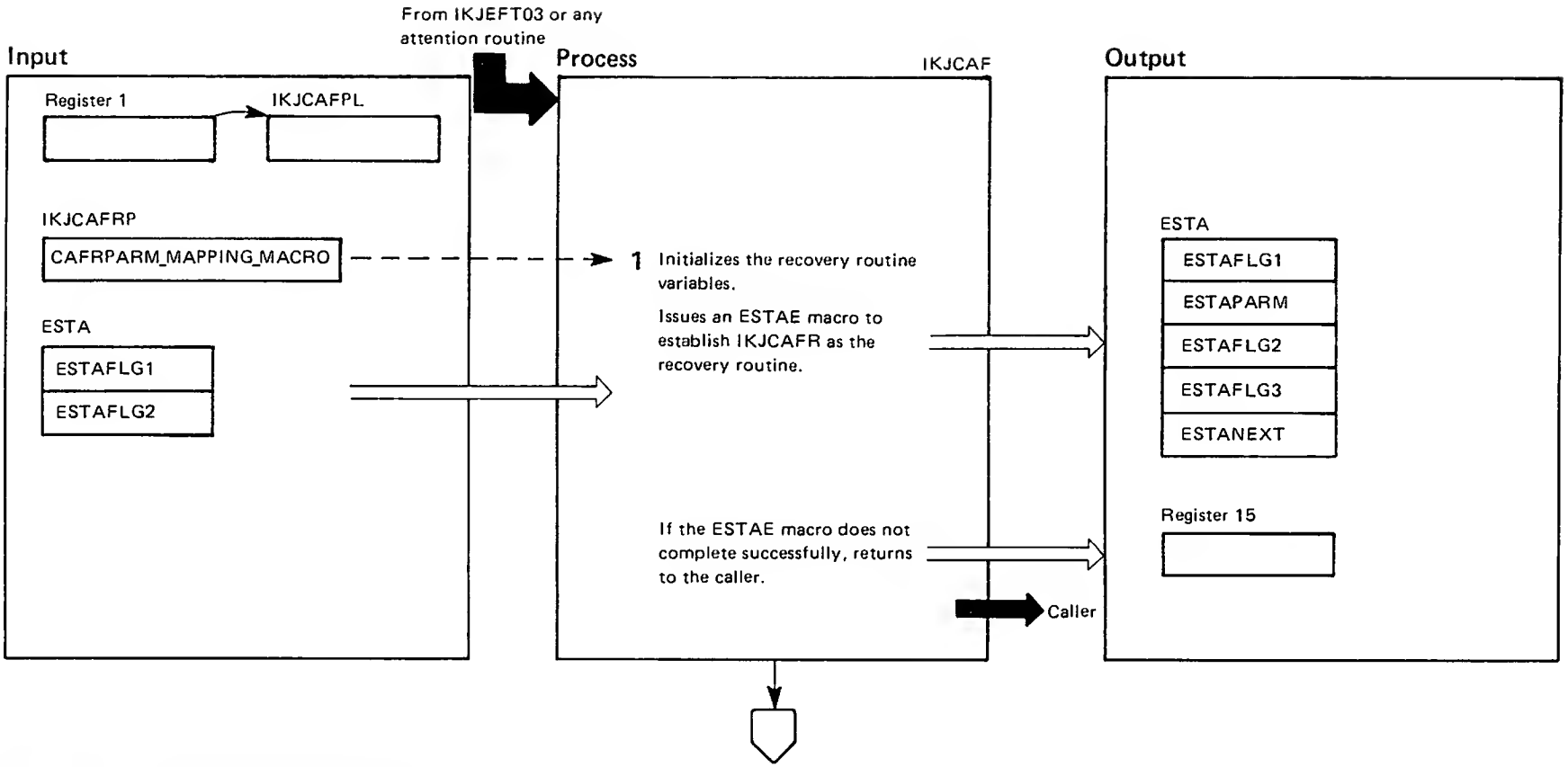


Diagram 23. CLIST Attention Facility (Part 1 of 10)

Key Description	Routine
At entry, register 1 points to the caller's parameter list. 1. Initializes the recovery routine variables. Issues an ESTAE macro to establish IKJCAFR as the recovery routine. If the ESTAE macro completes with a non-zero return code, returns to the caller with a return code of 20, indicating that an ESTAE exit could not be established.	IKJCAF

Diagram 23. CLIST Attention Facility (Part 2 of 10)

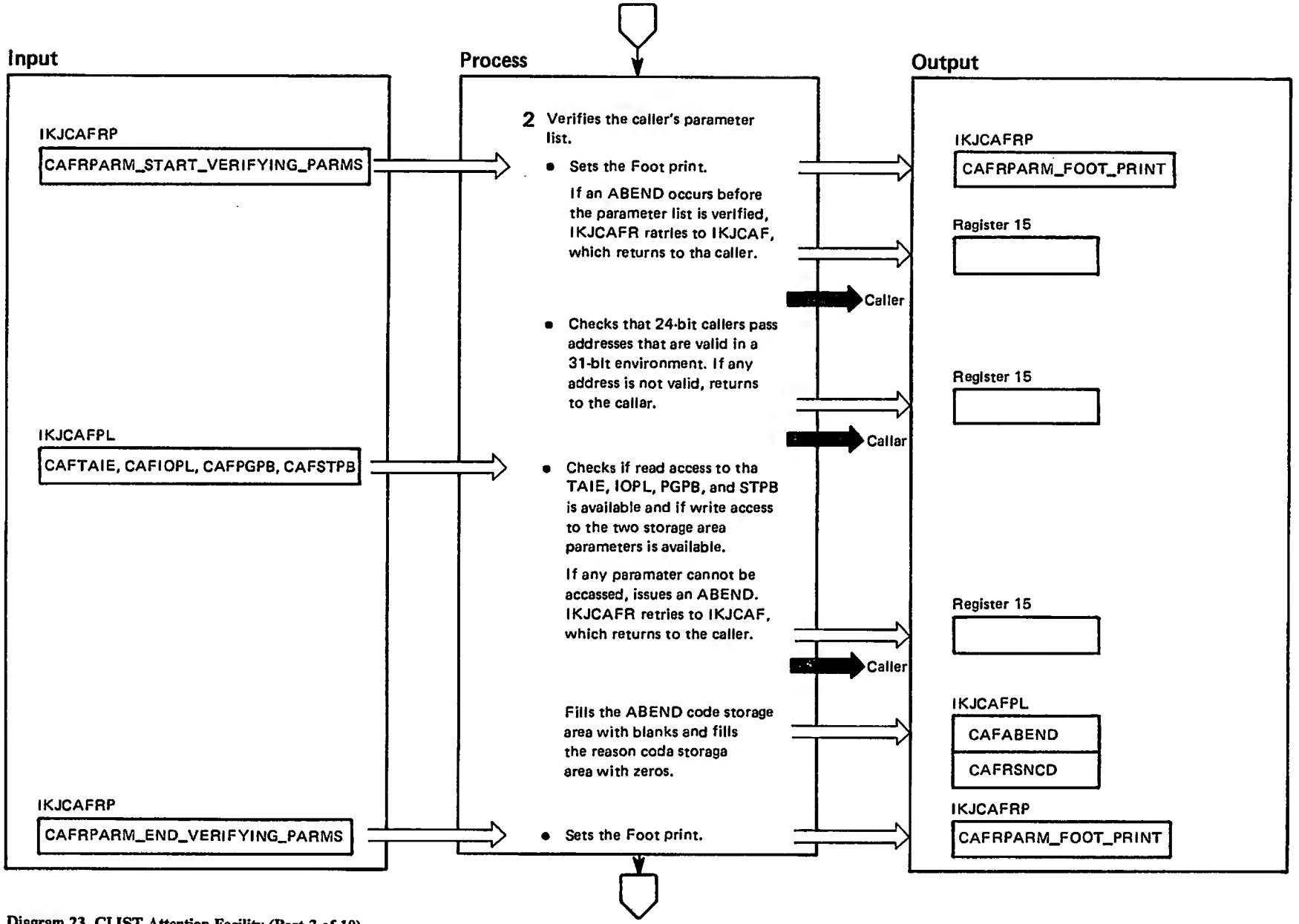


Diagram 23. CLIST Attention Facility (Part 3 of 10)

Key Description	Routine
2. Sets the footprint to indicate to IKJCAFR that IKJCAF is verifying the caller's parameter list. If an ABEND occurs before the parameter list can be verified, IKJCAFR receives control and retries to IKJCAF, which returns to the caller with a return code of 16. If the caller is a 24-bit program or function, verifies that the high-order bit of each of the caller's parameters is zero. If any high-order bit is not zero, returns to the caller with a return code of 28. Checks if it has read access to the TAIE, IOPL, PGPB, and STPB parameters and write access to the storage parameters. If it cannot access any parameter, an ABEND occurs and IKJCAFR receives control. IKJCAFR retries to IKJCAF, which returns to the caller with a return code of 24. Fills the ABEND code storage area with blanks and fills the reason code storage area with zeros. Sets the footprint to indicate that the caller's parameter list has been verified.	IKJCAF

Diagram 23. CLIST Attention Facility (Part 4 of 10)

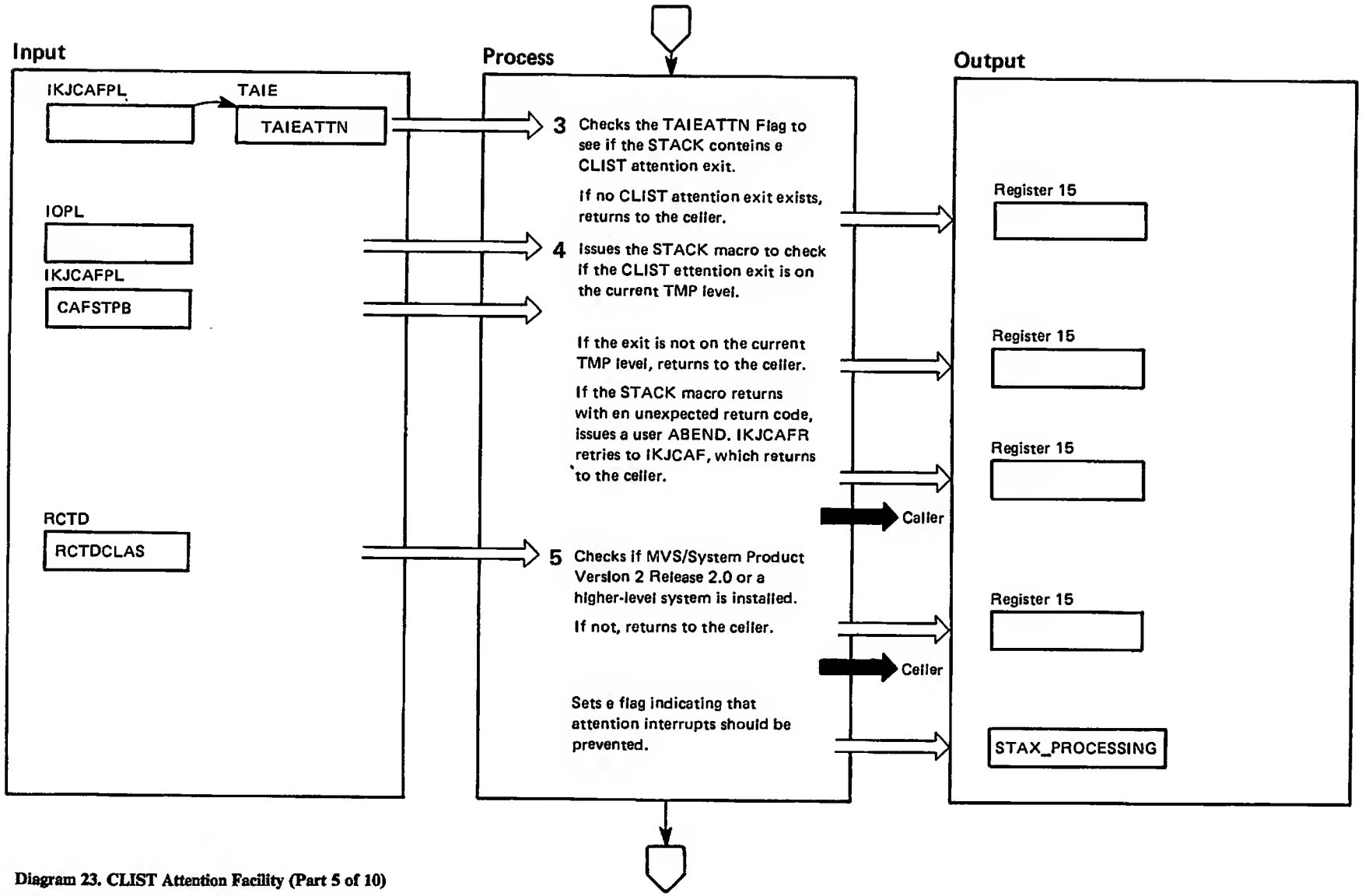


Diagram 23. CLIST Attention Facility (Part 5 of 10)

Key Description	Routine
3. Checks the TAIE flag to see if the STACK contains a CLIST attention exit. If the TAIE flag is off, returns to the caller with a return code of 8, indicating that no CLIST attention exit is to be processed. 4. Issues the STACK macro with the INQUIRE operand to check if the CLIST attention exit is on the current TMP level (the TMP can have multiple levels of parallel sides). If the exit is not on the current TMP level, returns to the caller with a return code of 8. If the STACK macro returns with an unexpected return code, issues a user ABEND of 601 and passes control to IKJCAFR. IKJCAFR retries to IKJCAF, which returns to the caller with a return code of 16. The return code from the STACK macro is passed to the caller as a reason code in the parameter list. 5. Checks if MVS/System Product Version 2 Release 2.0 or a higher-level system is installed. If the system requirement is not met, returns to the caller with a return code of 4. Sets the STAX processing flag to indicate that attention interrupts should be prevented.	IKJCAF

Diagram 23. CLIST Attention Facility (Part 6 of 10)

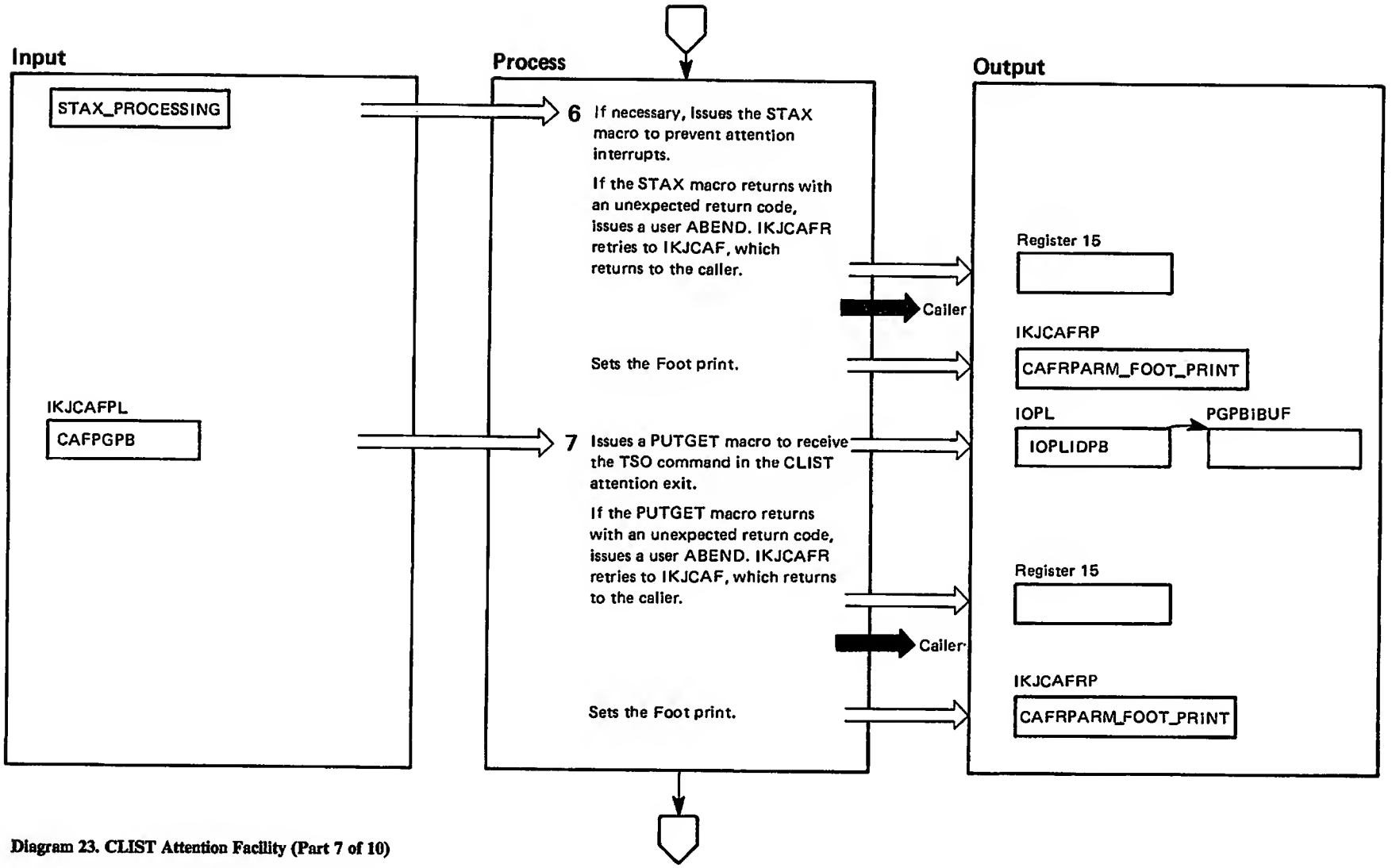


Diagram 23. CLIST Attention Facility (Part 7 of 10)

Key Description	Routine
6. Checks the STAX processing flag to see if the STAX macro must be issued to prevent attention interrupts. If so, issues the STAX macro with the IGNORE=YES operand. The STAX macro passes a return code to IKJCAF that indicates whether or not the caller already issued the STAX macro. IKJCAF later uses this return code to determine if it must issue the STAX macro to reestablish attention interrupts. If the STAX macro returns with an unexpected return code, issues a user ABEND of 600 and passes control to IKJCAFR. IKJCAFR retries to IKJCAF, which returns to the caller with a return code of 16. The return code from the STAX macro is passed to the caller as a reason code in the parameter list. Sets the foot print to indicate that attention interrupts are ignored. 7. Issues a PUTGET to the STACK to receive the TSO command coded in the CLIST attention exit. If the PUTGET macro returns with an unexpected return code, issues a user ABEND of 602 and passes control to IKJCAFR. IKJCAFR retries to IKJCAF, which returns to the caller with a return code of 16. The return code from the PUTGET macro is passed to the caller as a reason code in the parameter list. Sets the footprint to indicate that the PUTGET macro completed successfully.	

Diagram 23. CLIST Attention Facility (Part 8 of 10)

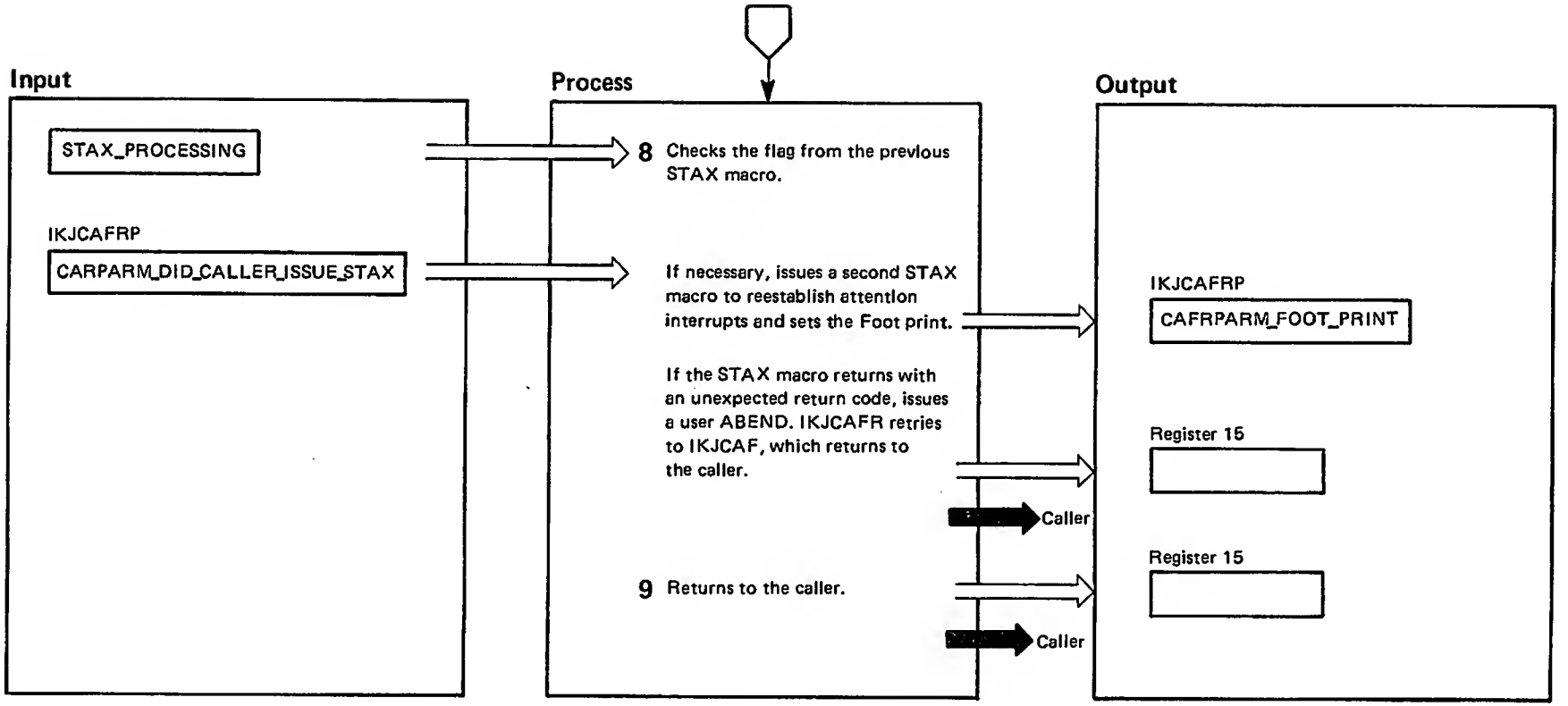


Diagram 23. CLIST Attention Facility (Part 9 of 10).

Key Description	Routine
8. Checks the flag from the previous STAX macro. If the caller did not prevent attention interrupts before invoking IKJCAF, issues a second STAX macro with the IGNORE=NO operand to reestablish attention interrupts and sets the footprint accordingly. If the STAX macro returns with a non-zero return code, issues a user ABEND of 600 and passes control to IKJCAFR. IKJCAFR retries to IKJCAF, which returns to the caller with a return code of 16. The return code from the STAX macro is passed to the caller as a reason code in the parameter list. 9. Upon normal completion, returns to the caller with a return code of zero.	

Diagram 23. CLIST Attention Facility (Part 10 of 10)

Program Organization

This section describes the organization of the CLIST attention facility.

Program Hierarchy

The CLIST attention facility consists of the IKJCAF load module, which is comprised of two routines:

IKJCAF - The CLIST attention facility mainline routine processes a CLIST with a CLIST attention exit and establishes the CLIST attention facility recovery routine.

IKJCAFR - The CLIST attention facility recovery routine processes any ABEND that occurs under the mainline routine.

Module Operation

The following descriptions outline the processing operations of the CLIST attention facility routine and its recovery routine.

IKJCAF -- CLIST Attention Facility Routine

- Set up recovery environment.
- Set footprints for recovery.
- Verify user parameter list.
- Check the TAIEATTN flag to make sure there is a CLIST attention exit to process.
- Issue STACK to determine if the CLIST attention exit is under the current level of the TMP.
- If a CLIST attention exit is present, check to make sure MVS SP 2.2.0 is installed.
- Issue STAX with the IGNORE=YES operand to prevent attention interrupt processing.
- Issue PUTGET to process the CLIST attention exit.
- Issue STAX with the IGNORE=NO operand to reestablish attention interrupt processing.
- Cancel recovery environment.
- Return to caller.

IKJCAFR -- CLIST Attention Facility Recovery Routine

- Check for SDWA; if none, percolate.
- If there is an SDWA, save registers in the passed save area.
- Set up addressability to the SDWA.
- Issue the SETRP macro.
- Fill in the SDWA.
- If the ABEND did not occur while checking the user parameter list then fill in the rest of the SDWA.
- Set up addressability to the VRA and initialize the VRA address.
- Fill in the VRA.
- If a retry has already been attempted and failed, place a comment in the VRA indicating that the retry attempt failed.
- Obtain the ABEND and reason codes:
  - Save the ABEND and reason codes from the SDWA to be passed back to the invoker of IKJCAF.
  - If the ABEND reason code is valid, pass it back to IKJCAF. Otherwise, pass back zeros.
- Check the footprint to determine the type of dump to issue:
  - If the ABEND occurred while verifying the user parameter list, issue the SETRP macro to indicate no dump is to be taken and retry to IKJCAF.
  - Otherwise, if the user is running authorized, issue an SDUMP.
  - If the user is not running authorized or the SDUMP is not successful, check the footprint to determine if the SETRP macro is to be issued to indicate percolate or retry. (A dump will be taken in either case.)
  - If the SDUMP is successful, check the footprint to determine if the SETRP macro is to be issued to indicate percolate or retry. (No dump will be taken.)
- Restore registers in the passed save area.
- Return to IKJCAF.

Directory

This table contains information to help you find the appropriate module operation description or assembly listing.

Label	Common Name	Load Module	Assembly Module	Control Section	Description
IKJCAF	CLIST Attention Facility Mainline Routine	IKJCAF	IKJCAF	IKJCAF	Processes a CLIST with a CLIST attention exit and establishes the recovery routine.
IKJCAFR	CLIST Attention Facility Recovery Routine	IKJCAFR	IKJCAFR	IKJCAFR	Processes any ABEND that occurs under the mainline routine.

Diagnostic Aids

This section contains a summary of the ABEND and return codes, and their meanings.

Routine	Hexadecimal Return Code	Meaning
IKJCAF	00	Normal completion.
	04	MVS/System Product Version 2 Release 2.0 is not on the system. The CLIST attention facility could not continue processing.
	08	The current attention interrupt is not for a CLIST with a CLIST attention exit.
	16	The CLIST attention facility issued an ABEND and retried.
	20	The CLIST attention facility could not establish an ESTAE exit. No processing occurred. *Note:* IKJCAF does not pass the return code from the ESTAE to the caller.
	24	The CLIST attention facility received invalid parameters from the calling program or function and retried.
	28	A 24-bit caller passed a parameter list that was invalid in the CLIST attention facility's 31-bit environment. (The high-order bit of an address was not zero.)

	Decimal ABEND Code	Meaning
	600	An issued STAX failed while trying to prevent attention interrupts. A reason code of xx (where xx is the return code from STAX) accompanies the user ABEND.
	601	An issued STACK failed while checking for a CLIST attention exit within a given STACK element. A reason code of xx (where xx is the return code from STACK) accompanies the user ABEND.
	602	An issued PUTGET failed while trying to receive the TSO command coded in a CLIST attention exit. A reason code of xx (where xx is the return code from PUTGET) accompanies the user ABEND.

Chapter 13. CLIST Processing and Diagnosis

This chapter is divided into two sections. The first section provides an overview of CLIST processing. CLIST diagnosis information follows the processing description.

CLIST Processing Overview

CLIST processing proceeds in two separate operations known as **Phase 1** and **Phase 2**. CLIST Phase 1 processing takes place under the EXEC command processor, which receives control from the terminal monitor program (TMP). CLIST Phase 2 processing takes place via the PUTGET service routine.

You may invoke the following exits during CLIST processing:

- CLIST initialization exit (IKJCT43I)
- CLIST termination exit (IKJCT43T)
- Built-In function exit (IKJCT44B)
- Installation-written statement exit (IKJCT44S).

For a description of these exits and when they are invoked during CLIST processing, see *TSO Extensions Customization*.

CLIST Phase 1 Processing

Phase 1 processing is done either explicitly or implicitly. The explicit form is defined when you enter 'EXEC' with a data set name; optionally, a value list (in quotes), and options. EXEC opens the data set explicitly named with the command and reads the records (CLIST statements and TSO/E commands) into storage.

The implicit form is defined when you enter only a member name (optionally preceded by a percent sign) of a partitioned CLIST library, optionally followed by a value list. However, the CLIST library must have been previously allocated to the file name 'SYSPROC' prior to the implicit EXEC command. Allocation could have been accomplished either by step allocation during LOGON or by using the TSO/E ALLOCATE command. When you enter an implicit EXEC command *without* the percent sign, EXEC does not get control immediately; the TMP first goes through its normal routine of searching a BLDL list to determine if a command exists. If the command name is not found in the BLDL list, the TMP invokes EXEC to determine if it is processing an implicit CLIST request.

The EXEC command determines if file name 'SYSPROC' is allocated and, if so, FINDs the member name specified. If there is no 'SYSPROC' data set allocated or if the FIND is unsuccessful, EXEC issues the message "COMMAND name-entered NOT FOUND". If the FIND is successful, EXEC reads the records (CLIST statements or TSO/E commands) from the specified member into storage. After the records are read into storage, the EXEC command builds an in-storage CLIST and places it on the stack to be processed by Phase 2.

If the EXEC command has been attached by another command as an implicit EXEC command and the member name entered could not be found as described by the two cases above, EXEC issues the message "SUBCOMMAND name-entered NOT FOUND".

CLIST Phase 2 Processing

CLIST Phase 2 processing takes place via the PUTGET service routine. When the PUTGET service routine is invoked, the GETLINE function of the I/O service routines receives control. GETLINE obtains the next line of data from the input stack. When GETLINE encounters a CLIST as the next line of data, it invokes CLIST Phase 2 processing. Phase 2 processes each statement within the CLIST that it recognizes as valid.

When CLIST Phase 2 processing encounters an unknown statement in a CLIST, it passes the statement back to the invoker of PUTGET for processing. To restart the CLIST, another PUTGET must be issued. Processing resumes, starting with the statement following the unknown statement. This process flow between PUTGET and CLIST Phase 2 repeats until all statements in the CLIST have been processed.

The CLIST symbolic substitution and variable access routine (IKJCT441) is a terminal I/O service routine invoked during CLIST Phase 2 processing. For a complete list of return codes and their meanings for IKJCT441, see the Diagnostic Aids section of the "Terminal I/O Service Routines" chapter in this book.

CLIST Diagnosis Information

This section describes information you can use to diagnose a CLIST problem. To find out how to locate this information in a dump, see *TSO Extensions System Diagnosis: Guide and Index.*

SDWA Contents

The following diagnostic information is placed in the system diagnostic work area (SDWA) during CLIST Phase 1 processing under the EXEC command processor:

SDWACSCT	–	The CSECT name in which the error occurred.
SDWAMODN	–	The load module name in which the error occurred.
SDWASC	–	The name of the subcomponent and the module subfunction in which the error occurred.
SDWAREXN	–	The name of the recovery routine handling the error.
SDWARRL	–	The label of the recovery routine that filled in this SDWA.
SDWACIDB	–	The component ID base (prefix) number.
SDWAMLVL	–	The level of the module in which the error occurred.

SDWAVRA Contents -- Phase 1

The following diagnostic information is placed in the variable recording area (VRA) during EXEC processing:

- The header "CLIST PHASE 1 PROCESSING".
- The descriptive name "ECDA" followed by the address of the EXEC common data area (ECDA) control block.
- The descriptive name "COMPROC" followed by the address of the command procedure (COMPROC) data area, if available.
- The descriptive name "FOOTPRINT BLOCK" followed by the address of the footprint data area.
- The descriptive name "COMMAND BUFFER" followed by as much of the command buffer as will fit in the remaining space of the VRA.

SDWAVRA Contents -- Phase 2

The following diagnostic information is placed in the variable recording area (VRA) during CLIST Phase 2 processing under the PUTGET service routine:

- The header "CLIST PHASE 2 PROCESSING".
- The descriptive name "LSD" followed by the address of the list source descriptor (LSD) control block.
- The descriptive name "EXD" followed by the address of the EXEC data (EXD) control block.
- The descriptive name "ECT" followed by the address of the environmental control table (ECT) control block.
- The descriptive name "COMPROC" followed by the address of the command procedure (COMPROC) data area, if available.

- The descriptive name "COMMON AREA" followed by the address of the common area control block.
- The descriptive name "FOOTPRINT BLOCK" followed by the address of the footprint data area.
- The descriptive name "STACK" followed by the address of the TSO/E I/O stack.

CLIST Dump Processing

CLIST processing performs the following dump actions:

- Fills in the VRADAE field to prevent duplicate dumps for the same symptoms.
- Checks for recursive abends and suppresses dumps and further execution at that point.
- Suppresses dumps for 'x37', 'x3E', '913', and 'x22' abends (except 122). Also, if an abend occurs during the execution of a data management macro while a CLIST is processing an I/O statement (OPENFILE, CLOSFILE, GETFILE, or PUTFILE), CLIST recovery will not issue a dump.
- Passes a SNAP macro parameter list to RTM in order to control the information that will be placed into a dump. The following options will be specified in the SNAP macro:

SDATA – Specifies the system control program information to be dumped:

– LSQA	The local system queue area.
– SWA	The scheduler work area related to the task.
– CB	The control blocks for the task.
– TRT	The GTF and system trace data.
– DM	Data management control blocks for the task.
– ERR	Recovery/termination control blocks for the task.
– PCDATA	Program call information.

PDATA – Specifies the problem program information to be dumped:

– PSW	Program status word.
– REGS	Floating and general registers.
– SA	Save area linkage information.
– JPA	Contents of the job pack area.
– LPA	Contents of the active link pack area.
– SPLS	All virtual storage symbols.
– SUBTASKS	The designated task and the program data information for all of its subtasks.

CLIST Footprint Description

CLIST processing provides a footprinting mechanism to determine which module was in control or which subfunction was executing at the time of an abend. Footprint information is maintained in a 48-byte (12 words) footprint data area that contains the CSECT name and module level for the executing module and for the module that executed previously. Figure 13-1 describes the footprint data area.

<8 bytes><16 bytes>

Current module CSECT	Current module level
Previous module CSECT	Previous module level

Figure 13-1. Footprint Data Area

The address of the footprint area is found in the VRA following the descriptive name "FOOTPRINT BLOCK". For more information about locating VRA information in a dump, see *TSO Extensions System Diagnosis: Guide and Index.*

CLIST Return Codes

CLIST processing sets the following return codes:

- 0 - Normal termination.
- 4 - CLIST was not found.
- 12 - Procedure was not executable.

Because CLIST processing propagates exit reason codes, you may receive other return codes than those listed. For a description of the reason and return codes issued by the standard format exits, see *TSO Extensions Customization.*

CLIST processing does not set any abend codes.

Services Used by CLIST Processing

CLIST processing uses the following routines during processing:

- IKJEFF02 - Message writing routine.
- IKJSCAN - To locate an implicit exec member name.
- IKJPARS - To syntax check explicit exec parms.
- IKJEFF19 - Analyze parse errors.
- IKJDAIR - Allocation services.
- IKJEFF18 - Analyze DAIR errors.

Chapter 2. Terminal Monitor Program

The terminal monitor program (TMP) obtains commands, gives control to command processors, and monitors their execution.

The IBM-supplied TMP is a program executed via the user logon procedure. The TMP mainline routine executes in either supervisor state or problem program mode. The TMP second-level routine always executes in problem program mode. The TMP mainline routine or CALL command processor attaches (as an APF-authorized subtask) any command processor whose name appears in the APF command table (module IKJEFTE2) or any program whose name appears in the APF program tables (IKJEFTE8 or IKJEFTAP). Commands and programs whose names are not in those tables are attached as non-APF-authorized subtasks of the TMP second-level routine or CALL command processor.

Note: The APF tables can be initialized and maintained using members of the PARMLIB data set. For information about using the PARMLIB data set, see *TSO Extensions Customization.*

The TMP obtains its first command from the EXEC PARM field and gives control to the appropriate command processor. After the first command has been processed, the TMP does one of the following:

- Obtains a new command and gives control to the appropriate command processor.
- Handles attention requests.
- Attempts to recover from errors in a command processor or one of its subtasks.
- Attempts to recover from errors in its own routines.
- Returns control to the LOGON/LOGOFF scheduler when the operator issues a STOP command or when the user enters a LOGON or LOGOFF command.

Index

A

ABEND codes
  for terminal monitor program 2-33
attention ECBs
  IOPLECB 2-13
attention requests (diagram) 2-12

C

catalog information routine 1-5
  diagnostic aids 7-7
  directory 7-6
  introduction 7-1
  method of operation 7-3
  program organization 7-6
CLIST attention facility 12-1
  diagnostic aids 12-17
  directory 12-16
  introduction 12-1
  method of operation 12-3
  module operation 12-14
  program organization
    program hierarchy 12-14
CLIST attention facility mainline routine (IKJCAF) 12-14
CLIST attention facility recovery routine (IKJCAFR) 12-14
CLIST diagnosis information 13-3
CLIST footprint information 13-5
CLIST Phase 1 processing 13-1
CLIST Phase 2 processing 13-1
CLIST processing 12-17
  EXEC command 13-1
  return codes 13-5
  services used by 13-5
CLIST statement processing
  CLIST attention facility 12-14
  LISTDSI modules 3-7
command processors 1-4
command scan modules
  command scan service routine (IKJEFP30) 4-20
  command scan syntax checking routine (IKJEFP40) 4-20
command scan service routine 1-5
  data areas 4-21
  diagnostic aids 4-25
  introduction 4-1
  method of operation 4-3
    command scan and parse service routines (overview) 4-4
    command scan service routine 4-6
  program hierarchy 4-18
command scan service routine *(continued)*
  program organization
    module operation 4-19—4-20

D

DAIR 1-5, 5-4
DAIRFAIL 1-5, 8-1
DAIRFAIL modules
  DAIR/SVC99 error code analyzer (IKJEFF18) 8-6
DAIR/SVC99 error code analyzer 1-5
  directory 8-7
  introduction 8-1
  method of operation 8-3
    analyzing DAIR/SVC99 error codes 8-4
  program organization
    module operation 8-6
    program hierarchy 8-6
default service routine 1-5
  diagnostic aids 6-7
  directory 6-6
  introduction 6-1
  method of operation 6-2
    default service routine 6-4
  program organization 6-6
double-byte character set (DBCS) support 4-19
dynamic allocation interface modules
  DAIR (IEFDB4D0) 5-6
dynamic allocation interface routine 1-5
  diagnostic aids 5-8
  directory 5-7
  introduction 5-1
  method of operation 5-3
    dynamic allocation interface routine 5-4
  program organization
    module operation 5-6
    program hierarchy 5-6

E

ESTAE requests (diagram) 2-16
ESTAI requests (diagram) 2-14

I

IEFDB4D0 5-6
IGX00035 10-4
IKJCAF 12-1, 12-14
IKJCAFR 12-14
IKJCTIOR 3-6
IKJCTTBL 3-7
IKJCT442 3-6

IKJEFF02 8-6, 9-1
IKJEFF18 1-5, 8-1, 8-6
IKJEFP00 4-19
IKJEFP03 4-19
IKJEFP05 4-19
IKJEFP06 4-18, 4-19
IKJEFP08 4-19
IKJEFP10 4-18
IKJEFP20 4-20
IKJEFP30 4-20
IKJEFP40 4-20
IKJEFP60 4-20
IKJEFTAP 2-1, 2-5, 2-18, 2-30
IKJEFTE2 2-30
IKJEFTE8 2-30
IKJEFTNS 2-30
IKJEFTPV 2-29
IKJEFTP1 2-28
IKJEFTP2 2-28
IKJEFTSC 2-28
IKJEFTSL 2-29
IKJEFTSR 1-6, 10-1
IKJEFT01 2-22
IKJEFT02 2-23
IKJEFT03 2-24
IKJEFT04 2-26
IKJEFT05 2-26
IKJEFT07 2-27
IKJEFT08 2-27
IKJEFT09 2-27
IKJEF02R 9-1
IKJEHCIR 1-5, 7-1
IKJEHDEF 1-5, 6-1
IKJLDI00 3-7
IKJLDI01 3-7
IKJLDI02 3-7
IKJLDI03 3-7
IKJLDI04 3-7
IKJLDI05 3-7
IKJLDI99 3-7
IKJPARSR 1-5, 4-4, 4-8, 4-10, 4-12, 4-18
IKJPARS2 service routine (IKJEFP60) 4-20
IKJPTGTR 3-6
IKJSCANR 1-5, 4-4, 4-6, 4-18
IKJTSLAR 1-3, 11-1
intialization
  of parse (diagram) 4-10
  of TMP (diagram) 2-6
introduction to TSO/E 1-1
IOPLECB 2-13

L

LAR
  See linkage assist routine
linkage assist routine
  for TMP 2-29
linkage assist routine *(continued)*
  for TSO/E services 1-3, 11-1
LISTDSI processing (IKJLDI00 - IKJLDI99) 3-7

M

method of operation diagrams
  analyzing DAIR/SVC99 error codes 8-4
  catalog information routine 7-4
  CLIST attention facility 12-4
  command scan and parse service routines (overview) 4-4
  command scan service routine 4-6
  default service routine 6-4
  dynamic allocation interface routine 5-4
  handling attention requests 2-12
  handling ESTAE requests 2-16
  handling ESTAI requests 2-14
  handling TSO commands with parallel TMP 2-10
  handling TSO commands with primary TMP 2-8
  parse initialization 4-10
  parse service routine 4-8
  primary TMP 2-4
  searching for IKJPARSR positional parameters 4-12
  searching for IKJPARS2 positional parameters 4-14
  searching for keyword parameters and subfields 4-16
  terminal I/O service routines (overview) 3-4
  TMP initialization 2-6
  TSO/E message issuer 9-4
  TSO/E service facility 10-2
  TSO/E service linkage assist routine 11-4
MVS/Extended Architecture considerations 1-2

P

PARMLIB tables 2-30
parse modules
  ECT setting routine (IKJEFP05) 4-19
  parse address PCE processor (IKJEFP03) 4-19
  parse mainline subroutines (IKJEFP08) 4-19
  parse messages (IKJEFP10) 4-18
  parse scan module (IKJEFP20) 4-20
  parse service routine (IKJEFP00) 4-19
  subroutines common to command scan and parse (IKJEFP06) 4-18, 4-19
parse service routine 1-5
  data areas 4-21
  diagnostic aids 4-25
  introduction 4-1
  method of operation 4-3
    command scan and parse service routines (overview) 4-4
    parse initialization 4-10
    parse service routine 4-8
    searching for IKJPARSR positional parameters 4-12

parse service routine *(continued)*
method of operation *(continued)*
searching for IKJPARS2 positional parameters 4-14
searching for keyword parameters and subfields 4-16
program organization
module operation 4-19—4-20
program hierarchy 4-18

R

return codes
for catalog information routine 7-7
for command scan 4-25
for default service routine 6-7
for dynamic allocation interface routine 5-8
for parse 4-25
for terminal I/O service routines 3-8
for terminal monitor program 2-33
for TSO service linkage assist routine 11-8

S

service routines
See TSO/E service routines

T

tables, PARMLIB 2-30
terminal I/O modules
terminal I/O service routine 1-5
diagnostic aids 3-8
introduction 3-1
list of modules 3-11
process overview 3-3
terminal I/O service routines (overview) 3-4
program organization
program hierarchy 3-6
services used by 3-11
terminal monitor program 1-1, 1-4
data areas 2-30
diagnostic aids 2-33
directory 2-31
introduction 2-1
method of operation 2-3
handling attention requests 2-12
handling ESTAE requests 2-16
handling ESTAI requests 2-14
handling TSO commands with parallel TMP 2-10
handling TSO commands with primary TMP 2-8
primary TMP 2-4
TMP initialization 2-6
program organization
control flow 2-19
module operation 2-22—2-29
program hierarchy 2-18
TMP
See terminal monitor program
TMP modules
attention exit routine (IKJEFT03) 2-24
call function routine (IKJEFT08) 2-27
ESTAE exit routine (IKJEFT05) 2-26
ESTAE retry routine (IKJEFT07) 2-27
ESTAI exit routine (IKJEFT04) 2-26
initialization routine (IKJEFT01) 2-22
initialization subroutines (IKJEFTP1) 2-28
linkage assist routine (IKJTSLAR) 2-29
mainline routine (IKJEFT02) 2-23
mainline stage 1 procedures (IKJEFTP2) 2-28
second-level routine (IKJEFT09) 2-27
service controller (IKJEFTSC) 2-28
service facility parameter verification (IKJEFTPV) 2-29
TSO Extensions considerations 1-3
TSO service routines
default service routine (IKJEHDEF) 6-4
TSOLNK
See IKJEFTSR
TSO/E commands
handling with parallel TMP (diagram) 2-10
handling with primary TMP (diagram) 2-8
TSO/E message issuer
directory 9-7
introduction 9-1
method of operation 9-3
TSO/E message issuer 9-4
program organization
module operation 9-6
program hierarchy 9-6
TSO/E service facility
diagnostic aids 10-8
directory 10-7
introduction 10-1
module operation 10-5
program organization
TSO/E service linkage assist routine 1-3
diagnostic aids 11-8
directory 11-7
entry points 11-3
introduction 11-1
method of operation 11-3
TSO/E service linkage assist routine 11-4
program organization
module operation 11-6
program hierarchy 11-6
TSO/E Service Routine 1-6
TSO/E service routine SVC 10-4
TSO/E service routines 1-5
catalog information routine (IKJEHCIR) 1-5, 7-4
CLIST attention facility (IKJCAF) 12-4
command SCAN routine 1-5
command scan routine (IKJSCANR) 4-4, 4-6
DAIR/SVC99 error code analyzer (IKJEFF18) 1-5, 8-4

TSO/E service routines *(continued)*
default service routine (IKJEHDEF) 1-5
dynamic allocation interface routine (DAIR) 1-5, 5-4
introduction 1-1
PARSE routine 1-5
parse routine (IKJPARSR) 4-4
parse service routine (IKJPARSR) 4-8, 4-10, 4-12
terminal I/O 1-5
TSO/E message issuer (IKJEF02R) 9-4
TSO/E service facility (IKJEFTSR) 10-2
TSO/E service linkage assist routine (IKJTSLAR) 11-4
TSO/E Service Routine (IKJEFTSR) 1-6
TSO/E system
command processors 1-4
introduction 1-1
MVS/XA considerations 1-2
service routines 1-5—1-6
terminal monitor program 1-1, 1-4
TSO/E considerations 1-3
user programs 1-4

U

user programs 1-4

TSO Extensions
Terminal Monitor
Program and Service
Routines

READER'S
COMMENT
FORM

LY28-1308-4

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

LY28-1308-4

Reader's Comment Form

Cut or Fold Along Line

Fold ond tope | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold ond tope | Please Do Not Staple | Fold and tape

Printed in U.S.A.

IBM®

Program Number
5665-285

File Number
S370-39

LY28-1308-4

Printed in U.S.A.